PRESERVING DATA INTEGRITY IN DISTRIBUTED SYSTEMS

Dissertation

zur Erreichung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät der
Humboldt-Universität zu Berlin

von
Herrn Diplom-Informatiker
Marvin Triebel

Präsidentin der Humboldt-Universität zu Berlin:
Prof. Dr.-Ing. Dr. Sabine Kunst

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät:
Prof. Dr. Elmar Kulke

Gutachter:

1. Prof. Dr. Wolfgang Reisig

2. Prof. Dr. Marco Montali

3. Prof. Dr. Mathias Weske

Tag der mündlichen Prüfung: 28.11.2018

ACKNOWLEDGEMENTS

Ada Euler, Amrei Störmer-Schnuppner, Anja Schupp-ner, Andre Moelle, Andrey Rivkin, Arthur Bartels, Birgit Heene, Birgit Schiefner, Bo Zhao, Carla Stock, Christian Gierds, David Karcher, Diana Walter, Diego Calvanese, David Karcher, Dirk Fahland, Dimitris Karagiannis, Esteban Pavese, Florian Furbach, Franziska Bathelt-Tok, Gabriele Graichen, Hendrik Hügel, Irmela Triebel, Jan Sürmeli, Jörg Desel, Jörgen Brandt, Johannes Schneider, Jonathan Bräuer, Julian Hölscher, Karsten Wolf, Kathrin Otte, Kim Völlinger, Lars Grunske, Leonie Czycykowski, Liana Liebmann, Lucas Sakizloglou, Malte Graubner, Marc Kewitz, Marcin Hewelt, Marco Montali, Matthias Kreuschner, Matthias Weidlich, Meike Zehlike, Michael Rücker, Nicholas Heintz, Nora Schindler, Ognjen Savkovic, Olivia Kaletta-Graubner, Oswald Bertholt, Paul-David Brotmann, Rafael Triebel, Richard Müller, Robert Prüfer, Roland Barth, Roland Meyer, Samira Akili, Sarah-Christin Mueller-Downs, Simon Heiden, Sinem Getir, Sophie Euler, Sophie Herrmann, Sophie Tätweiler, Stefan Sprenger, Stephan Mennicke, Stephan Pfeiler, Sven Strickroth, Thomas Vogel, Tobias Heintz, Torsten Liebke, Tim Jungnickel, Ulf Leser, Uwe Nestmann, Vincent Kyas, Werner Nutt, Wolfgang Reisig, Wolfgang Triebel, Yannic Noller, Yasmin Patzer

♡

THANK YOU.

ABSTRACT

Information systems process data that is logically and physically *distributed* over many locations. Data entities at different locations may be in a specific relationship. For example, a data entity at one location may contain a reference to a data entity at a different location, or a data entity may contain critical information such as a password. The semantics of data entities induce *data integrity* in the form of requirements. For example, no references should be dangling, and critical information should be available at only one location. Data integrity discriminates between correct and incorrect data distributions. If a system distributes data such that data integrity is violated, it is prone to errors.

A distributed system progresses in steps, which may occur concurrently. In each step, data is manipulated. Each data manipulation is performed locally and affects a bounded number of data entities. A distributed system *preserves* data integrity if each step of the system yields a data distribution that satisfies the requirements of data integrity. *Preservation of data integrity* is a necessary condition for the correctness of a system. Analysis and design are challenging, as distributed systems lack global control, employ different technologies, and data may accumulate unboundedly.

In this thesis, we study formal methods to model and analyze distributed data-aware systems. As a result, we provide a technology-independent framework for design-time analysis. To this end, we use *algebraic Petri nets*. We show that there exists a bound for the conditions of each step of a distributed system if and only if the steps can be described by a finite set of transitions of an algebraic Petri net. We use *algebraic equations* and *inequalities* to specify data integrity. We show that preservation of data integrity is undecidable in case we consider all reachable steps. We show that preservation of data integrity is decidable in case we also include unreachable steps. We show the latter by showing computability of a non-preserving step as a *witness*.

ZUSAMMENFASSUNG

Informationssysteme verarbeiten Daten, die logisch und physisch über viele Knoten *verteilt* sind. Datenobjekte verschiedener Knoten können dabei Bezüge zueinander haben. Beispielsweise kann ein Datenobjekt eine Referenz auf ein Datenobjekt eines anderen Knotens oder eine kritische Information wie ein Passwort enthalten. Die Semantik der Daten induziert *Datenintegrität* in Form von Anforderungen: Zum Beispiel sollte keine Referenz verwaist und kritische Informationen nur an einem Knoten verfügbar sein. Datenintegrität unterscheidet gültige von ungültigen, fehlerhaften Verteilungen der Daten. Wenn ein System Daten so verteilt, dass die Datenintegrität verletzt wird, ist es fehleranfällig.

Ein verteiltes System verändert sich in Schritten, die nebenläufig auftreten können. Jeder Schritt manipuliert Daten. Jede Manipulation erfolgt lokal und betrifft beschränkt viele Datenobjekte. Ein verteiltes System *erhält* Datenintegrität, wenn alle Schritte in einer Datenverteilung resultieren, die die Anforderungen von Datenintegrität erfüllen. Die *Erhaltung von Datenintegrität* ist daher ein notwendiges Korrektheitskriterium eines Systems. Der Entwurf und die Analyse von Datenintegrität in verteilten Systemen sind schwierig, weil ein verteiltes System nicht global kontrolliert werden kann, verschiedene Technologien eingesetzt werden und Daten unbeschränkt wachsen können.

In dieser Arbeit untersuchen wir formale Methoden für die Modellierung und Analyse verteilter Systeme, die mit Daten arbeiten. Wir entwickeln die Grundlagen für die Verifikation von Systemmodellen. Dazu verwenden wir *algebraische Petrinetze*. Wir zeigen, dass die Schritte verteilter Systeme mit endlichen vielen Transitionen eines algebraischen Petrinetzes beschrieben werden können, genau dann, wenn eine Schranke für die Bedingungen aller Schritte existiert. Wir verwenden algebraische Gleichungen und Ungleichungen, um Datenintegrität zu spezifizieren. Wir zeigen zum einen, dass die Erhaltung von Datenintegrität unentscheidbar ist, wenn alle erreichbaren Schritte betrachtet werden. Zum anderen zeigen wir, dass die Erhaltung von Datenintegrität entscheidbar ist, wenn auch unerreichbare Schritte berücksichtigt werden. Dies zeigen wir, indem wir die Berechenbarkeit eines nicht-erhaltenden Schrittes als *Zeugen* zeigen.

PUBLICATIONS OF THE AUTHOR

[TS16a] Marvin Triebel and Jan Sürmeli. "Homogeneous Equations of Algebraic Petri Nets." In: *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*. Ed. by Josée Desharnais and Radha Jagadeesan. Vol. 59. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 14:1–14:14.

[TS16b] Marvin Triebel and Jan Sürmeli. *Homogeneous Equations of Algebraic Petri Nets*. Tech. rep. 2016.

[TS16c] Marvin Triebel and Jan Sürmeli. "Characterizing Stable and Deriving Valid Inequalities of Petri Nets." In: *Fundam. Inform.* 146.1 (2016), pp. 1–34.

[TS15] Marvin Triebel and Jan Sürmeli. "Characterizing Stable Inequalities of Petri Nets." In: *Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings*. Ed. by Raymond R. Devillers and Antti Valmari. Vol. 9115. Lecture Notes in Computer Science. Springer, 2015, pp. 266–286.

[ST13a] Jan Sürmeli and Marvin Triebel. "Synthesizing Cost-Minimal Partners for Services." In: *Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings*. Ed. by Samik Basu et al. Vol. 8274. Lecture Notes in Computer Science. Springer, 2013, pp. 584–591.

[ST13b] Jan Sürmeli and Marvin Triebel. *Cost-optimizing compositions of services - analysis and synthesis*. Informatik-Berichte 242. Humboldt-Universität zu Berlin, 2013.

CONTENTS

LIST OF DEFINITIONS

LIST OF THEOREMS, LEMMAS AND COROLLARIES

LIST OF FIGURES

1 INTRODUCTION

1.1 BACKGROUND

Computational systems are *distributed* for different reasons. Often, it is an inherent property of the system, where different actors with different objectives interact, such as a buyer and a seller. Then, data is necessarily distributed. A computational system may also be distributed to achieve availability and robustness. Here, redundancy and references may provide high availability, which a centralized architecture cannot provide due to physical limitations [GL02]. Moreover, monolithic rigidity creates a single point of failure. When an error occurs in a distributed system, only the affected component shuts down and other components may continue processing [Alv+16]. To achieve privacy and security of a computational system, data is distributed over many components. In that scenario, different data at different locations is not a burden, but a requirement [Bie+16; Han+04].

If data is distributed, data entities at different locations may be in a specific relationship. For example, a data entity may contain a reference to a data entity at a different location. Alternatively, a data entity may contain sensitive information, such as a password. The relationship of data entities induce requirements on the distribution of data. For example, every reference has to be valid or a password is never copied among multiple locations of the system. Requirements may be motivated by data quality, security, or privacy. *Data integrity* consists of a set of requirements, which distinguishes valid from invalid, flawed distributions of data. This notion of data integrity generalizes the concept of data integrity known from classical databases [AHV95].

A distributed system evolves in steps. A step may be local or an interaction, such as a message between actors. If the interaction is flawed, a step may violate data integrity and the system is prone to errors. Restoring from errors during run time is expensive and data loss may be irreducible. The requirements of data integrity are expected to be *preserved* among every step. Designing the interaction of components of a distributed system is challenging: Different components use different technologies [Van+17], have different objectives and resources [Lyn96; Rei98]. Moreover, communication is asynchronous, actions are performed concurrently and each action is

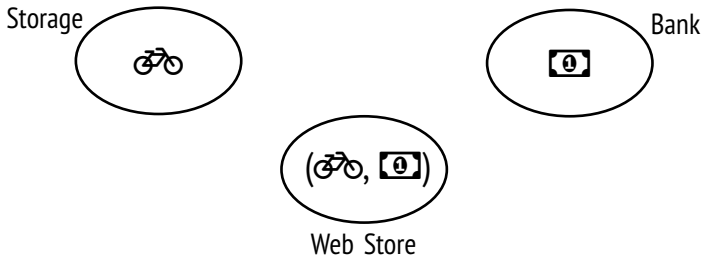


Figure 1.: A purchase order of a web store with distributed data.

performed with physically bounded bandwidth. Distributed systems require diligent engineering.

EXAMPLE. A short example of distributed data is sketched in Figure 1. The example shows three locations: Storage, Bank, and Web Store. Whereas a product 🚲 is stored in the Storage, the money 💵 is stored in the Bank. The Web Store contains a purchase order, which is the pair (🚲, 💵), consisting of a product and a price. Here, the purchase order contains a reference to the product 🚲 in the Storage. The money is a critical resource, which only the Bank should have access to. If the purchase order refers to a bicycle that is not in the Storage or the money is in the Storage and not on the Bank, the requirements of data integrity are not satisfied. For example, removing the product 🚲, without removing the purchase order (🚲, 💵) would be a step of the system that does not preserve data integrity.

Formal Methods

This thesis is built on the assumption that understanding the mathematical properties of an engineering problem helps to find a solution to the problem. Hence, we apply *formal methods*. In the following, we briefly motivate the methods applied in this thesis.

Technologies for computational machines such as programming languages and data models change frequently. In polyglot computing, it is a desired property to design a distributed system using varying, multiple technologies [Van+17]. Here, a sufficiently abstract modeling technique is necessary. *Formal modeling* describes mathematically based techniques for the modeling, specification, and development of systems. Additionally, in the field of formal methods the analysis and verification of formal models are studied, as well as the limitations such as unsolvable problems and relations to other modeling techniques. A *verification problem* consists of a system and a specification. The question is whether the system satisfies the specification.

However, for analysis purposes the answer to this question may be insufficient. It may also be necessary to understand *why* and *where* a specification is satisfied or violated. A *witness* can serve this purpose [McC+11]. A witness is a mathematical artifact that implies the satisfaction or violation of the given specification of a system while being reasonably simple.

The result of this thesis is not a finished software product. However, we bridge the gap between formal modeling and system design by presenting *prototypes*.

1.2 PROBLEM STATEMENT

In this thesis, we study how formal methods may provide answers to the following questions:

- How can we ensure that a distributed system preserves data integrity?
- How can we detect design flaws that violate data integrity in distributed systems?

It is well known that in many scenarios these problems are infeasible [Las16; Min67; Bag+13; RF10] from a verification point-of-view. Hence, it is part of both questions to identify problems, which are infeasible by any computing machine. Understanding the reasons of infeasibility helps engineers to identify potential sources of complexity for the design and analysis of systems. Engineering tools for verification results in a trade-off situation: On the one hand, using an expressive modeling language helps engineers to describe the systems appropriately. On the other hand, it is desired to study conditions under which analysis and verification is possible.

1.3 CONTRIBUTIONS

In this thesis, we provide four contributions. The first two contributions fall in the field of modeling of data integrity in distributed data-aware systems. The third and fourth contribution are with respect to the analysis and verification of data integrity in distributed data-aware systems.

MODELING. We contribute to the field of modeling of data integrity in distributed systems in the following points:

1. We study the known concept of algebraic Petri nets [Rei91] for the suitability of modeling data-aware distributed systems. We show in Theorem 16 that the steps of a distributed data-aware system can be described by a finite set of transitions if and only if the set of steps is monotone, bounded and structured.

2. As a second step, we study the known concept of algebraic equations and inequalities [Hac72; Rei13; Vau86] for the specification of data integrity. We study differences to other notions of data integrity [AHV95; DMM15]. Furthermore, we embed the theory of algebraic equations and inequalities into related work on the data-aware modeling of systems, and the specification and verification of a data-aware formal models.

ANALYSIS. We contribute to the field of analysis and verification of the preservation of data integrity in the following points. Both contributions are based on algebraic Petri nets over *compact data schemes*, where a finite representation of all *unifiers* of each *unification problem* [BS94] is computable and for each *dis-unification problem dis-unifiability* [BBM16] is decidable.

3. We show in Theorem 44 that preservation of an algebraic equation or inequality is undecidable in an algebraic Petri net in case we consider all reachable steps.

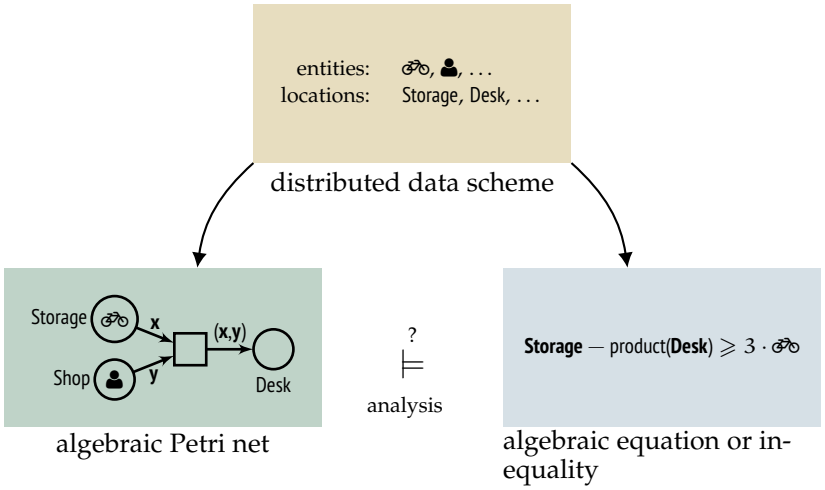
4. We show in Theorem 103 that the preservation of an algebraic equation or inequality is decidable in case we also include unreachable steps. We show this by showing computability of a non-preserving step.

1.4 OVERVIEW OF THE THESIS

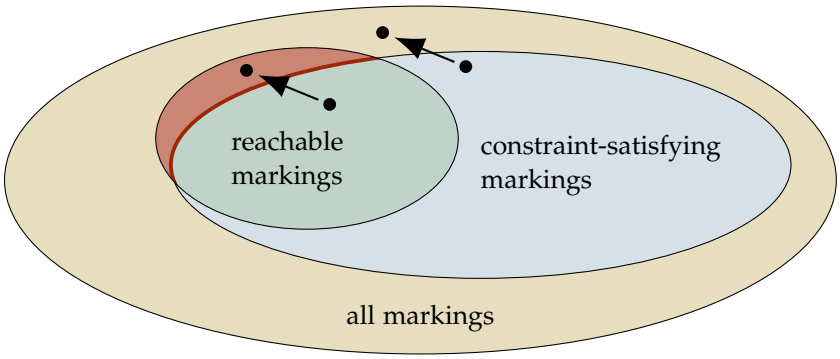
The thesis is split into three parts and an appendix. We summarize each separately.

Part 1: Data Integrity in Distributed Systems

In Part I, we study formal methods for data integrity in distributed systems. The formal framework is visualized in Figure 2. In Chapter 2, we use *algebraic Petri nets* [Rei91] for the modeling of distributed data-aware system. We first study *distributed data schemes* to model distributed data. We use the quotient algebra of an algebraic specification [ST12] to define *data schemes* modeling data entities. We identify the relevant subclass of *compact* data schemes, where the set of



(a) Modeling data integrity in distributed systems.



(b) Non-Preserving steps.

Figure 2.: Overview of the thesis.

unifiers of each *unification problem* is *finitely representable* [BS94] and the emptiness of the set of *dis-unifiers* of each *dis-unification problem* [BBM16] is decidable. Based thereon, a data scheme equipped with a set of *places* is a *distributed data scheme* and induces the set of *markings* and *transitions* [Rei91]. In Theorem 16, we show that a set of steps is *monotone*, *bounded*, and *structured* if and only if it can be modeled by a finite set of *transitions* of an *algebraic Petri net*. In Chapter 3, we study specification of *data integrity* by *algebraic equations* and *inequalities* [Rei13; Vau86]. We integrate the presented framework into related work on the data-aware modeling of systems. In Chapter 4, we study the analysis and verification of the preservation of data integrity and show in Theorem 44, that the preservation of an algebraic equation or inequality is undecidable in case we consider all reachable steps. In Chapter 5, we use an example to illustrate the modeling and analysis of the preservation of data integrity in a distributed system.

Part 2: Computing Non-Preserving steps

In Part II, we prove Theorem 103: computability of non-preserving steps if the underlying data scheme is *compact*, and unreachable steps are included. We give an overview of the proof in Chapter 6, reduce from general transitions to *simple transitions* in Chapter 7, study the solutions of *Diophantine equations* and *inequalities* in Chapter 8, show computability of a *linear representation* of the set of solutions of an *algebraic equation* or *inequality* in Chapter 9, and show the computability of non-preserving steps from a linearly represented set of markings in Chapter 10.

Part 3: Closure

We present two prototypical case studies to evaluate the computational approach of part II in Chapter 11. We conclude the thesis and discuss open questions and future work in Chapter 12.

Appendix

The appendix includes a brief summary of the mathematical notations, an index, lists of definitions, lemmas, corollaries, theorems, figures, and the bibliography.

Part I.

Data Integrity in Distributed Systems

In part I, we present our formal framework. In Chapter 2, we study *algebraic Petri nets* to model *distributed data-aware systems*. In Chapter 3, we study *algebraic equations* and *inequalities* to specify *data integrity*. In Chapter 4, we study analysis and verification of *preservation* of algebraic equations and inequalities in algebraic Petri nets. In Chapter 5, we use an example to illustrate the modeling and analysis of data integrity in a distributed system.

2 DISTRIBUTED DATA-AWARE SYSTEMS

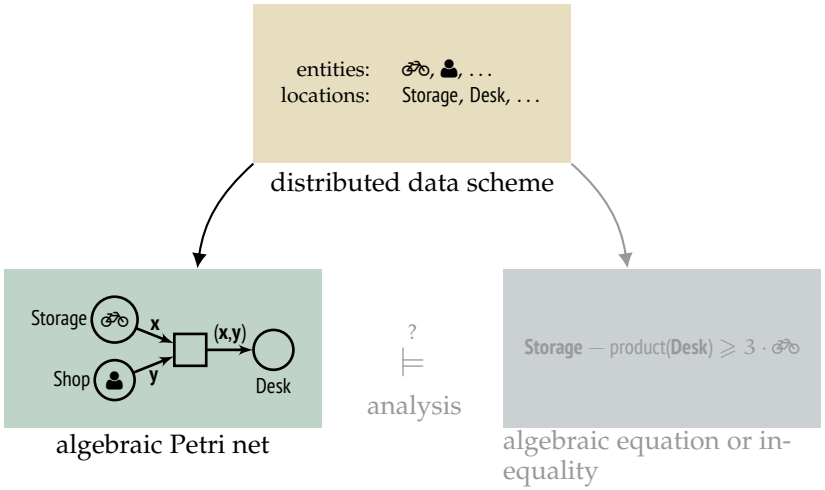


Figure 3.: Overview of part II. The contents of chapter 3 and 4 are grayed out.

In this chapter, we formalize distributed data-aware systems. To this end, we recall the definition of *algebraic Petri nets* [Rei91]. Figure 3 shows an overview of part I. We first study the statics of a distributed data-aware system. We formalize distributed data by means of *markings* of a *distributed data scheme* in Section 2.1. We continue in Section 2.2 with the modeling of dynamics of a distributed data-aware system by means of *transitions* of an algebraic Petri net. The specification and analysis of distributed data integrity is grayed out in Figure 3, as it will be discussed in Chapters 3 and 4.

In this chapter, we build on established theoretical foundations. Important references are mentioned in the text. The chapter ends with a detailed discussion of related work.

2.1 DISTRIBUTED DATA SCHEMES

In this section, we recall how to model the statics of a distributed data-aware system with *markings* [Rei13].

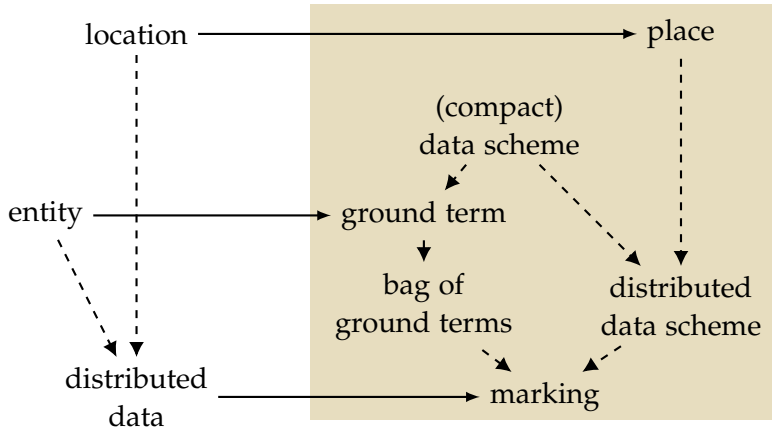


Figure 4.: Overview of concepts. Formalizations introduced in this section are highlighted in sand color. Concepts motivated in Chapter 1 are shown on the left. Horizontal arrows show formalizations, dashed arrows show dependencies in the definitions.

We introduce the terminology shown in Figure 4. On the left side, the concepts motivated in Chapter 1 are shown. On the right side the definitions that formalize these concepts are shown and highlighted in sand color. Horizontal arrows depict formalizations. Dashed arrows depict dependencies in the definitions. We first model *entities* and operations on entities as *ground terms* of a *data scheme* in Section 2.1.1. A data scheme is known as a *quotient algebra* of an *algebraic specification* [ST12]. We identify the relevant subclass of *compact* data schemes, where the set of *unifiers* of each *unification problem* is *finitely representable* [BS94] and the emptiness of the set of *dis-unifiers* of each *dis-unification problem* [BBM16] is decidable.

In Section 2.1.2, we study how a relational model can be embedded in a data scheme. The *tuple-product* of data schemes builds a bridge to relational algebra and preserves *compactness*, comparable to [BS92].

In Section 2.1.3, we extend data schemes with *places*, that formalize locations, resulting in *distributed data schemes*. Finally, we recall *markings* [Rei13] for modeling distributed data. *Parameterized effects* generalize markings.

We assume that the reader is familiar with the notions shown in Figure 5. The used notations are listed in the second column, the definitions can be found in the appendix, Chapter A.

2.1.1 Data Scheme

In this section, we recall how to model data entities as ground terms [ST12]. To this end, we first define a data scheme. Intuitively, a data

notion	notation	definition
set of function symbols	$\mathbb{F}$	A.4
set of variables	$\mathbb{V}$	A.4
set of terms ($F \subseteq \mathbb{F}$, $V \subseteq \mathbb{V}$)	$\langle {V \atop F} \rangle$	A.4
set of ground terms ($F \subseteq \mathbb{F}$)	$\langle {\emptyset \atop F} \rangle$	A.4
set of substitutions ($V, W \subseteq \mathbb{V}$, $F \subseteq \mathbb{F}$)	$V \overset{F}{\rightsquigarrow} W$	A.4
axiom of terms ($\theta, \theta' \in \langle {\mathbb{V} \atop F} \rangle$)	$\theta \doteq \theta'$	A.6
term congruence induced by a set of axioms A	$\equiv_A$	A.4, A.6
bags of terms ($F \subseteq \mathbb{F}, V \subseteq \mathbb{V}$)	$\mathbb{N}\langle {V \atop F} \rangle$	A.5
set of realizations of a term/substitution X	$\mathcal{R}(X)$	A.6

Figure 5.: Notions and notations used in this section.

scheme consists of a finite set of function symbols $F \subseteq \mathbb{F}$, where $\mathbb{F}$ is the universe of all function symbols over arbitrary sorts. F induces its *Herbrand structure* over the ground terms $\langle {\emptyset \atop F} \rangle$. Each ground term models an entity. The second ingredient of a data scheme is a *congruence* on the Herbrand structure, which is an equivalence relation that is preserved by application of all functions $f \in F$. The congruence can be seen as the semantics of the ground terms, which are syntactical objects: Some terms may be interpreted as the same entity, which is expressed as a congruence. The approach is known as the *quotient algebra* of an *algebraic specification* [ST12].

Terms, the Herbrand structure, and term congruences are defined in Section A.4.

> **Definition 1** (Data Scheme)
> Let $F \subseteq \mathbb{F}$ be a finite set of function symbols, and $\equiv \subseteq \langle {\emptyset \atop F} \rangle \times \langle {\emptyset \atop F} \rangle$ be an F-congruence.
>
> Then, $(F, \equiv)$ is a *data scheme*. We denote the *ground terms of* $(F, \equiv)$ by $\langle {\emptyset \atop F} \rangle$.

A set of axioms A can be used to describe $\equiv_A$, the least congruence containing all realizations of the axioms. The definition can be found in Section A.6.

We illustrate Definition 1 using the data schemes shown in Figure 6. The first data scheme, $\left(F_{6.1}, \equiv_{A_{6.1}}\right)$, models the natural numbers. The intuition is that the entities are the natural numbers: Starting from 0 and enumerating all its successors. So, each ground term models a natural number. Here, two function symbols are used: 0 with arity

function symbols	axioms terms	
$\mathsf{F}_{6.1}$	$\mathsf{A}_{6.1}$	$\langle {}^{\emptyset}_{\mathsf{F}_{6.1}} \rangle$
$0 : N$	-	0, $\mathsf{suc}(0)$,
$\mathsf{suc} : N \to N$		$\ldots$
$\mathsf{F}_{6.2}$	$\mathsf{A}_{6.2}$	$\langle {}^{\emptyset}_{\mathsf{F}_{6.2}} \rangle$
$0 : N$	$\mathbf{x}{+}0 \doteq \mathbf{x}$	$0{+}0$,
$\mathsf{suc} : N \to N$	$\mathbf{x}{+}\mathsf{suc}(\mathbf{y}) \doteq \mathsf{suc}(\mathbf{x}{+}\mathbf{y})$	$0{+}\mathsf{suc}(0)$,
$+ : N \times N \to N$		$\ldots$
$\mathsf{F}_{6.3}$	$\mathsf{A}_{6.3}$	$\langle {}^{\emptyset}_{\mathsf{F}_{6.3}} \rangle$
$0 : N$	$\mathsf{odd}(0) \doteq \mathsf{false}$	0, $\mathsf{suc}(0)$,
$\mathsf{suc} : N \to N$	$\mathsf{odd}(\mathsf{suc}(0)) \doteq \mathsf{true}$	true, false,
$\mathsf{true} : B$	$\mathsf{odd}(\mathsf{suc}(\mathsf{suc}(\mathbf{x}))) \doteq \mathsf{odd}(\mathbf{x})$	$\mathsf{odd}(0)$,
$\mathsf{false} : B$		$\mathsf{odd}(\mathsf{suc}(0))$,
$\mathsf{odd} : N \to B$		$\ldots$

Figure 6.: Three data schemes modeling natural numbers using sort N: $(\mathsf{F}_{6.1}, \equiv_{\mathsf{A}_{6.1}})$ without addition, $(\mathsf{F}_{6.2}, \equiv_{\mathsf{A}_{6.2}})$ with addition, and $(\mathsf{F}_{6.3}, \equiv_{\mathsf{A}_{6.3}})$ with a second sort B for Boolean values.

zero and suc with arity one. The signature of the symbols are all over the same sort, N. The ground terms in $\langle {}^{\emptyset}_{\mathsf{F}_{6.1}} \rangle$ are 0, $\mathsf{suc}(0)$, etc. As $\mathsf{A}_{6.1} = \emptyset$, we have $\equiv_{\mathsf{A}_{6.1}} = \equiv_{\emptyset}$, which reduces to equality. Hence, two ground terms are equivalent if and only if they are equal.

The data scheme $(\mathsf{F}_{6.2}, \equiv_{\mathsf{A}_{6.2}})$ extends $(\mathsf{F}_{6.1}, \equiv_{\mathsf{A}_{6.1}})$ by an additional binary function $+$. We use infix notation for $+$. $\mathsf{A}_{6.1}$ contains two axioms. The congruence induced by $\mathsf{A}_{6.2}$ specifies the semantics of addition. For example, $0{+}0$ and 0 are equivalent. Intuitively, both ground terms model the entity zero. The reason is as follows: If we map $\mathbf{x}$ to 0, we see that the tuple $(0{+}0, 0)$ is a realization of the first axiom.

$(\mathsf{F}_{6.3}, \equiv_{\mathsf{A}_{6.3}})$ extends $(\mathsf{F}_{6.1}, \equiv_{\mathsf{A}_{6.1}})$ with a second sort, B, modeling Boolean values. The data scheme has two symbols with arity zero of sort B: true and false. Furthermore, odd is a unary function from N to B. Intuitively, odd maps to true for all ground terms that model odd natural numbers. For example, we have $\mathsf{odd}(\mathsf{suc}(\mathsf{suc}(0))) \equiv_{\mathsf{A}_{6.3}} \mathsf{false}$.

As we will discuss in Section 2.3, we omit specifying the sorts explicitly. In our approach, the terms are build over the sorts used

$B_{7.1}$: $[0, 0]$	$B_{7.2}$: $[0, 0{+}0]$

Figure 7.: Two equivalent bags of ground terms of the second data scheme $\left(F_{6.2}, \equiv_{A_{6.2}}\right)$ modeling populations of of natural numbers.

by the function symbols of the data scheme. In Figure 6 this is the sort N for $\left(F_{6.1}, \equiv_{A_{6.1}}\right)$ and $\left(F_{6.2}, \equiv_{A_{6.2}}\right)$, and N and B for $\left(F_{6.3}, \equiv_{A_{6.3}}\right)$. By omitting the explicit definition of sorts we avoid notational overhead. However, the explicit modeling of sorts could be implemented as "syntactic sugar". Another design choice of this thesis is to avoid arbitrary interpretations of function symbols. We restrict ourselves to the Herbrand structure. However, we consider arbitrary congruences. This is equivalent to considering arbitrary interpretation, which are *junk-free* [ST12], as we will discuss in Section 2.3. An interpretation contains junk, if an element of an interpretation is not represented by a term.

We model a population of entities as a *bag of ground terms of* F and lift congruences accordingly. The notations and definitions used for bags can be found in Section A.3. Each ground term is mapped to a natural number and only finitely many ground terms are mapped to a value greater zero.

Definition 2 (Bags of Ground Terms)
Let $(F, \equiv)$ be a data scheme. Then, $\mathbb{N}\langle^{\emptyset}_{F}\rangle$ denotes the *set of all bags of ground terms of* F.

Furthermore, let for two bags of ground terms $B, B' \in \mathbb{N}\langle^{\emptyset}_{F}\rangle$ and all $\theta \in \langle^{\emptyset}_{F}\rangle$ hold the following:

$$\sum_{\substack{\theta' \in \langle^{\emptyset}_{F}\rangle \\ \theta' \equiv \theta}} B(\theta') = \sum_{\substack{\theta' \in \langle^{\emptyset}_{F}\rangle \\ \theta' \equiv \theta}} B'(\theta')$$

Then, B and B' are *equivalent with respect to* $\equiv$, written $B \equiv B'$

In Figure 7, two bags of ground terms of the data scheme $\left(F_{6.2}, \equiv_{A_{6.2}}\right)$ of Figure 6 are shown: $B_{7.1}$ and $B_{7.2}$. Again, $\left(F_{6.2}, \equiv_{A_{6.2}}\right)$ models the natural numbers with addition. In Figure 7, we use the following notation for bags: Each ground term in the bag is shown as often as the number assigned to the ground term by the bag, enclosed by brackets. We observe that $B_{7.1}$ and $B_{7.2}$ are equivalent: As $0{+}0 \equiv_{A_{6.2}} 0$, we have:

$$B_{7.1}(0) + B_{7.1}(0{+}0) = 1 + 1 = 2 = B_{7.2}(0)$$

unification problem		representation of all unifiers	
$E_{8.1}$:	$(\ \mathbf{x}+0\ ,\ \mathrm{suc}(0)\ )$	$\Omega_{8.1}$:	$\mathbf{x} \mapsto \mathrm{suc}(0)$
$E_{8.2}$:	$(\ \mathbf{x}+0\ ,\ 0+\mathrm{suc}(\mathbf{y})\ )$ $(\ \mathbf{y}\ ,\ \mathrm{suc}(0)\ )$	$\Omega_{8.2}$:	$\mathbf{x} \mapsto \mathrm{suc}(\mathrm{suc}(0))$ $\mathbf{y} \mapsto \mathrm{suc}(0)$
$E_{8.3}$:	$(\ \mathbf{x}\ ,\ \mathrm{suc}(\mathbf{y})\ )$	$\Omega_{8.3}$:	$\mathbf{x} \mapsto \mathrm{suc}(\mathbf{y})$ $\mathbf{y} \mapsto \mathbf{y}$
$E_{8.4}$:	$(\ \mathbf{x}\ ,\ \mathrm{suc}(\mathbf{y})\ )$ $(\ \mathbf{y}\ ,\ \mathrm{suc}(\mathbf{x})\ )$	$\Omega_{8.4}$:	$\emptyset$

Figure 8.: Four unification problems over the data scheme $\left(\mathsf{F}_{6.2}, \equiv_{\mathsf{A}_{6.2}}\right)$ of Figure 6, each with a representation of all unifiers.

We observe that for a data scheme $(\mathsf{F}, \equiv)$, the lifted relation $\equiv \subseteq \mathbb{N}\langle \frac{\emptyset}{\mathsf{F}} \rangle \times \mathbb{N}\langle \frac{\emptyset}{\mathsf{F}} \rangle$ is an equivalence relation.

In the following, we identify the class of *compact data schemes*, where the set of *unifiers* of each *unification problem* [BS94] is finitely representable and for each *dis-unification problem* [BS92] it is decidable whether it is *dis-unifiable*.

UNIFICATION. Given a data scheme and a finite set of pairs of terms, a unification problem asks whether for each tuple both terms can be unified to equivalent ground terms by a *substitution*. The definition of a substitution can be found in Section A.4. Intuitively, a substitution replaces variables in terms by other terms.

Definition 3 (Unification Problem, Unifier, Unifiable)
Let $(\mathsf{F}, \equiv)$ be a data scheme. Let $\mathsf{E} \subseteq \langle \frac{\mathbb{V}}{\mathsf{F}} \rangle \times \langle \frac{\mathbb{V}}{\mathsf{F}} \rangle$ be finite.

Then, E is a *unification problem of* $(\mathsf{F}, \equiv)$.

Let $\sigma \in \mathbb{V}(\mathsf{E}) \overset{\mathsf{F}}{\rightsquigarrow} \emptyset$ such that for all $(\theta, \theta') \in \mathsf{E}$:

$$\sigma(\theta) \equiv \sigma(\theta')$$

Then, σ is a *unifier of* E. We denote *set of all unifiers of* E by $\mathbb{U}(\mathsf{E})$. E is *unifiable* if $\mathbb{U}(\mathsf{E}) \neq \emptyset$.

In Figure 8, four unification problems are shown. The first unification problem, $E_{8.1}$ consists of one pair of terms, where only the left term $\mathbf{x}+0$ contains the variable $\mathbf{x}$. The terms are unifiable because there exists a term θ such that replacing $\mathbf{x}$ with θ results in a ground term that

is equivalent to suc(0): suc(0)+0) and suc(0) are equivalent and θ can be chosen as suc(0). Accordingly, the substitution in $\Omega_{8.1}$, that maps **x** to suc(0), is a unifier. Moreover, all other unifiers are equivalent. For example choosing $\theta = 0$+suc(0) would also result in a unifier. Hence, the third column is a representation of all unifiers for the first unification problem.

In the second row of Figure 8, the unification problem $E_{8.2}$ with two pairs and two variables, **x** and **y**, is shown. The second pair, (**y**+0, suc(0)) implies that every unifier assigns suc(0) (or an equivalent term) to **y**. Combining this observation with the first pair, (**x**+0, 0+suc(**y**)), we deduce that every unifier assigns suc(suc(0)) to **x** (or an equivalent term). Hence, $\Omega_{8.2}$ is a representation of $\mathbb{U}(E_{8.2})$.

Contrary to $E_{8.1}$ and $E_{8.2}$, the third unification problem, $E_{8.3}$, has infinitely many unifiers, which are not all equivalent. For example, if 0 is assigned to **y** and suc(0) is assigned to **x**, the terms **x** and suc(**y**) result in the term suc(0). However, replacing **y** with any term θ and replacing **x** with suc(θ') results in a unifier. Hence, $\Omega_{8.3}$ is a representation of all unifiers using the variable **y**.

The fourth unification problem is not unifiable and thus $\Omega_{8.4} = \emptyset$ is a representation of all unifiers. The intuition is the following: The first pair, (**x**, suc(**y**)), states that **x** is the successor of **y** and the second pair, (**y**, suc(**x**)), states that **y** is the successor of **x**. It is not possible to find terms for **x** and **y**, such for each of both pairs the terms become equivalent.

DIS-UNIFICATION. Symmetrically to a unification problem, we define a *dis-unification problem*. Here, we restrict ourselves to the case that the right term is always ground. A dis-unification problem is *dis-solvable* if there exists a substitution such that all pairs of terms are not equivalent.

Definition 4 (Dis-Unification Problem, Dis-Unifier, Dis-Unifiable)
Let $(F, \equiv)$ be a data scheme. Let $E \subseteq \langle {}^{\mathbb{V}}_{F} \rangle \times \langle {}^{\emptyset}_{F} \rangle$ be finite.

Then, E is a *dis-unification problem of* $(F, \equiv)$.

Let $\sigma \in \mathbb{V}(E) \overset{F}{\rightsquigarrow} \emptyset$ such that for all $(\theta, \theta') \in E$:

$$\sigma(\theta) \not\equiv \theta'$$

Then, σ is a *dis-unifier of* E. We denote *set of all dis-unifiers of* E by $\not\mathbb{U}(E)$. E is *dis-unifiable* if $\not\mathbb{U}(E) \neq \emptyset$.

Figure 9 illustrates Definition 4. Two dis-unification problems are shown: $E_{9.1}$ and $E_{9.2}$. Both contain two tuples. The first dis-unification

dis-unification problem		dis-unifiable?
$\mathsf{E}_{9.1}$:	$\{ \mathsf{odd}(\mathbf{x}) \;, \mathsf{odd}(0) \}$ $\{ \mathsf{odd}(\mathsf{f}(\mathsf{f}(\mathbf{x})) \;, \mathsf{odd}(0) \}$	✓
$\mathsf{E}_{9.2}$:	$\{ \mathsf{odd}(\mathbf{x}) \;, \mathsf{odd}(0) \}$ $\{ \mathsf{odd}(\mathsf{f}(\mathbf{x})) \;, \mathsf{odd}(0) \}$	×

Figure 9.: Two dis-unification problems over the data scheme $\left(\mathsf{F}_{6.3}, \equiv_{\mathsf{A}_{6.3}}\right)$.

problem $\mathsf{E}_{9.1}$ is dis-unifiable. If we choose a substitution σ with $\sigma(\mathbf{x}) = \mathsf{f}(0)$ we have $\mathsf{odd}(\mathsf{f}(0)) \not\equiv_{\mathsf{A}_{6.3}} \mathsf{odd}(\mathsf{f}(0))$ and $\mathsf{odd}(\mathsf{f}(\mathsf{f}(\mathsf{f}(0)))) \not\equiv_{\mathsf{A}_{6.3}} \mathsf{odd}(\mathsf{f}(0))$.

The second dis-unification problem $\mathsf{E}_{9.2}$ is not dis-unifiable. Intuitively, for every natural number either the number or its successor is odd. Accordingly, we cannot find a substitution for $\mathbf{x}$ such that $\mathbf{x}$ and $\mathsf{f}(\mathbf{x})$ are even.

COMPACT DATA SCHEMES. A data scheme is *compact*, if two properties are satisfied. First, for every unification problem a finite representation of the set of all unifiers is computable. Second, it is decidable if all dis-unification problems are dis-unifiable. In [BS94], a data scheme, where all unification problems are finitely representable are referred to as a *finitary equational theory*. Here, we apply the notions of representation and realization, denoted by $\mathcal{R}(\cdot)$, as defined in Section A.6. A ground term θ is a realization of a term θ', if θ is a ground term obtained by replacing variables by ground terms in θ'.

> **Definition 5** (Compact Data Scheme)
> Let $(\mathsf{F}, \equiv)$ be a data scheme. Let for all unification problems E of $(\mathsf{F}, \equiv)$ hold:
>
> - There exists a finite and computable set of substitutions $\Omega \subseteq \mathbb{V}(\mathsf{E}) \overset{\mathsf{F}}{\rightsquigarrow} \mathbb{V}$ such that: $\mathbb{U}(\mathsf{E}) = \mathcal{R}(\Omega)$.
>
> Let for all dis-unification problems E hold:
>
> - Emptiness of $\not\mathbb{U}(\mathsf{E})$ is decidable.
>
> Then, $(\mathsf{F}, \equiv)$ is *compact*.

From a pure modeling point-of-view, the distinction between data schemes and compact data schemes is not of great interest. On the contrary, for analytical questions and verification approaches, this distinction is fundamental. Hence, for all decidability results regarding

analysis as described in Chapter 4 and Part II, it is a prerequisite that the model is built using a compact data scheme. However, we restrict ourselves from the possibility to analyze arbitrary data structures. We will discuss this restriction in Section 2.3.

2.1.2 Relational Algebra in Data Schemes

In this section, we build a bridge to relational algebra using formal reasoning. In the theory of relational algebra a tuple is an entity. We observe that it is possible to model tuples and relations using data schemes.

Intuitively, a tuple can be seen as a special function symbol *tuple*. The resulting sort is a fresh sort that is not used in any of the other data schemes. On the resulting sort we introduce a new function symbol for projection: $\mathit{projection}_i$ to project to the i-th member.

Definition 6 (Tuple-Product of Data Schemes)
Let $(F_1, \equiv_1)$ and $(F_2, \equiv_2)$ be data schemes. Let $s_1 \in \text{Sorts}(F_1)$ and $s_2 \in \text{Sorts}(F_2)$. Let $t \in \mathbb{S} \setminus (\text{Sorts}(F) \cup \text{Sorts}(F'))$, and $\mathit{tuple} \in \mathbb{F}$ with $\text{Signature}(\mathit{tuple}) = s_1 \times s_2 \to t$. Let $\mathit{projection}_1 \in \mathbb{F}$ with $\text{Signature}(\mathit{projection}_1) = t \to s_1$. Let $\mathit{projection}_2 \in \mathbb{F}$ with $\text{Signature}(\mathit{projection}_2) = t \to s_2$. Let $x_1 \in \mathbb{V}$, $x_2 \in \mathbb{V}$ with $\text{Sort}(x_1) = s_1$ and $\text{Sort}(x_2) = s_2$. Let

$$
\begin{aligned}
F' = \{ & \mathit{tuple} : s_1 \times s_2 \to t, \\
& \mathit{projection}_1 : t \to s_1 \\
& \mathit{projection}_2 : t \to s_2 \} \\
A = \{ & \mathit{projection}_1(\mathit{tuple}(x_1, x_2)) \doteq x_1, \\
& \mathit{projection}_2(\mathit{tuple}(x_1, x_2)) \doteq x_2 \}
\end{aligned}
$$

Let $F = F_1 \cup F_2 \cup F'$ and $\equiv$ be the least F-congruence with $(\equiv_1 \cup \equiv_2 \cup \equiv_A) \subseteq \equiv$.

Then, $(F, \equiv)$ is the *tuple-product of* $(F_1, \equiv_1)$ *and* $(F_2, \equiv_2)$ *over* s_1, s_2.

In Figure 10a two data schemes and a product of them is shown. The data scheme $(F_{10.1}, \equiv_{A_{10.1}})$ models a customer 👤 as entity. $F_{10.1}$ uses one sort C and $\langle \emptyset_{F_{10.1}} \rangle$ is singleton. The second data scheme, $(F_{10.2}, \equiv_{A_{10.2}})$, models products: A bicycle 🚲, and a car 🚗. The third data scheme, $(F_{10.3}, \equiv_{A_{10.3}})$ is a tuple-product of $(F_{10.1}, \equiv_{A_{10.1}})$ and $(F_{10.2}, \equiv_{A_{10.2}})$ over C and P. The induced equivalence on the entities fits the intuition of the projection functions prj_1 and prj_2. An

function symbols	axioms	terms
$F_{10.1}$: 👤 : C	$A_{10.1}$: -	$\langle {}^{\emptyset}_{F_{10.1}} \rangle$: 👤
$F_{10.2}$: 🚗 : G 🚲 : G	$A_{10.2}$: -	$\langle {}^{\emptyset}_{F_{10.2}} \rangle$: 🚗, 🚲
$F_{10.3}$: 👤 : C 🚗 : G 🚲 : G tuple : $G \times C \to T$ prj_1 : $T \to G$ prj_2 : $T \to C$	$A_{10.3}$: $\text{prj}_1(\text{tuple}(\mathbf{x},\mathbf{y})) \doteq \mathbf{x}$ $\text{prj}_2(\text{tuple}(\mathbf{x},\mathbf{y})) \doteq \mathbf{y}$	$\langle {}^{\emptyset}_{F_{10.3}} \rangle$: 👤, 🚗, 🚲, tuple(👤,🚲), tuple(👤,🚗), prj_1(tuple(👤,🚲)), ...

(a) Two data schemes and their product using C, G, T as sorts and **x**,**y** as variables.

$$\text{prj}_1(\text{tuple}(👤, 🚗)) \equiv_{A_{10.3}} 👤$$
$$\text{prj}_2(\text{tuple}(👤, 🚲)) \equiv_{A_{10.3}} 🚲$$

(b) Equivalences of the data scheme product.

Figure 10.: The tuple-product of two data schemes.

example is shown in Figure 10b, where one pair is projected to the first member and one pair is projected to the second member.

Definition 6 is restricted to pairs. However, if we chain the product we can model tuples of arbitrary length.

In the following lemma, we observe that the product of two compact data schemes is again a compact data scheme under two assumptions: First, the sets of sorts of each set of function symbols is disjoint, and second, the congruence is specified by a finite set of axioms.

Lemma 7 (Compactness of Tuple-Product)
Let $(F_1, \equiv_1)$ and $(F_2, \equiv_2)$ be compact data schemes with:

1. $\textsc{Sorts}(F_1) \cap \textsc{Sorts}(F_2) = \emptyset$,
2. For $i = 1, 2$, there exists finite sets of axioms $A_i \subseteq \langle {}^{\mathbb{V}}_{F_i} \rangle \times \langle {}^{\mathbb{V}}_{F_i} \rangle$ with $\equiv_{A_i} = \equiv_i$.

Let $(F_{1\cdot2}, \equiv_{1\cdot2})$ be a tuple-product of $(F_1, \equiv_1)$ and $(F_2, \equiv_2)$.
Then, $(F_{1\cdot2}, \equiv_{1\cdot2})$ is a compact data scheme.

Proof of Lemma 7. Let E be a unification or dis-unification problem of $(F_{1\cdot2}, \equiv_{1\cdot2})$ with

$$\begin{aligned} F' = \{ & \mathit{tuple} : s_1 \times s_2 \to t, \\ & \mathit{projection}_1 : t \to s_1 \\ & \mathit{projection}_2 : t \to s_2 \} \\ A = \{ & \mathit{projection}_1(\mathit{tuple}(x_1, x_2)) = x_1, \\ & \mathit{projection}_2(\mathit{tuple}(x_1, x_2)) = x_2 \} \end{aligned}$$

W.l.o.g we assume $\textsc{Sort}(\theta) = \textsc{Sort}(\theta')$ for all $(\theta, \theta') \in E$. Otherwise, E is not unifiable and dis-unification can be obtained by removing all tuples of unequal sort.

The proof is structured as follows: We encode E into a set of (dis-)unification problems of $(F_1, \equiv_1)$ and $(F_2, \equiv_2)$ and such that each (dis-)unifier can be translated to a (dis-)unifier of E.

Each ground term of sort t is of the form $\mathit{tuple}(\eta_1, \eta_2)$. Hence, we replace every variable $x \in \mathbb{V}_t$ by the term $\mathit{tuple}(x_1, x_2)$ for fresh variables $x_1 \in \mathbb{V}_{s_1}$ and $x_2 \in \mathbb{V}_{s_2}$ in E resulting in E'. We observe that E is unifiable (dis-unifiable) if and only if E' is unifiable (dis-unifiable). Moreover, every unifier σ' of E' can be translated into a unifier σ of E by defining:

$$\sigma(x) = \begin{cases} \mathit{tuple}(x_1, x_2) & \text{, if } x \in \mathbb{V}_t \\ x & \text{, otherwise.} \end{cases}$$

For $i \in \{1, 2\}$, every term in E' that contains the function symbol $\mathit{projection}_i$ is of the form $\mathit{projection}_i(\mathit{tuple}(\theta_1, \theta_2)$ with $\theta_i \in \langle {}^{\emptyset}_{F_{1\cdot2}} \rangle$. By the axioms, we have $\mathit{projection}_i(\mathit{tuple}(\theta_1, \theta_2) \equiv_{1\cdot2} \theta_i$. Thus, we replace every occurrence of $\mathit{projection}_i(\mathit{tuple}(\theta_1, \theta_2)$ by θ_i resulting in E''. By the equivalence, the of unifiers or dis-unifiers of E' and E'' are the same.

Now, every tuple in E'' that contains the function symbol *tuple* is of the form $\big(\mathit{tuple}(\zeta_1, \zeta_2), \mathit{tuple}(\zeta_1', \zeta_2')\big)$ with $\zeta_1, \zeta_1' \in \langle {}^{\mathbb{V}}_{F_1} \rangle$ and $\zeta_2, \zeta_2' \in \langle {}^{\mathbb{V}}_{F_2} \rangle$.

Then, for each remaining $\big(\mathit{tuple}(\zeta_1, \zeta_2), \mathit{tuple}(\zeta_1', \zeta_2')\big) \in E$, we have $\zeta_1, \zeta_1' \in \langle {}^{\mathbb{V}}_{F_1} \rangle$ and $\zeta_2, \zeta_2' \in \langle {}^{\mathbb{V}}_{F_2} \rangle$. All other tuples are either terms of F_1 oder terms of F_2. Now, we distinguish the following cases:

1st case: E is a unification problem.
We consider

$$E''' = \Big(E'' \setminus \{\big(tuple(\zeta_1, \zeta_2), tuple(\zeta_1', \zeta_2')\big)\}\Big) \cup \{(\zeta_1, \zeta_1'), (\zeta_2, \zeta_2')\}$$

And observe that E'' and E' have the same set of (dis-)unifiers, as $\sigma(tuple(\zeta_1, \zeta_2)) \equiv_{1.2} \sigma(tuple(\zeta_1', \zeta_2'))$ if and only if $\sigma(\zeta_1) \equiv_1 \sigma(\zeta_1')$ and $\sigma(\zeta_2) \equiv_2 \sigma(\zeta_2')$. Then, we have for all tuples $\big(\theta, \theta'\big) \in E'''$ that $\theta, \theta' \in \langle {}^{\mathbb{V}}_{F1} \rangle$ or $\theta, \theta' \in \langle {}^{\mathbb{V}}_{F2} \rangle$. Hence, we define for $i \in \{1, 2\}$:

$$E_i = \left\{ \big(\theta, \theta'\big) \in E''' \mid \theta, \theta' \in \langle {}^{\mathbb{V}}_{F_i} \rangle \right\}$$

As the variables of E_1 and E_2 are disjoint, we may treat each unification separately. Moreover, E_i is a unification problem of $(F_i, \equiv_i)$ and thus a finite representation of all unifiers is computable and can be combined to a finite representation of all unifiers of E.

2nd case: E is a dis-unification problem.
Each $\big(tuple(\zeta_1, \zeta_2), tuple(\zeta_1', \zeta_2')\big)$ is dis-unifiable if and only if (ζ_1, ζ_1') or (ζ_2, ζ_2') is dis-unifiable. Accordingly, we consider for $i \in \{1, 2\}$ the dis-unification problem:

$$E_i'' = \Big(E \setminus \{\big(tuple(\zeta_1, \zeta_2), tuple(\zeta_1', \zeta_2')\big)\}\Big) \cup \{(\zeta_i, \zeta_i')\}$$

We can apply this construction inductively and obtain a finite set of dis-unifications problems $\mathcal{E}$ such that for each $E''' \in \mathcal{E}$ we have that all tuples are either in F_1 or from F_2. Then, they are dis-unifiable if and only id each respective subset if dis-unifiable. As $(F_i, \equiv_i)$ is compact for $i \in \{1, 2\}$, we deduce that for each $E''' \in \mathcal{E}$ it is decidable whether E'' is dis-unifiable. Thus, it is decidable if E is dis-unifiable.

In the following chapters, our computability results are constrained to compact data schemes. Lemma 7 implies that all our decidability results carry over to products of compact data schemes. Many examples used in the following chapters rely on the product of compact data schemes. Hence, it is ensured that the resulting data scheme is a compact data scheme for all examples using tuples. Other classes of compact data schemes can be found in related work [BS94; BBM16; Sch86; Sch89]. For example, commutativity axioms and Herbrand structures are covered.

2.1.3 Distributed Data Scheme

In this section, we extend data schemes as defined in the last section for the distributed setting resulting in *distributed data schemes*. We recall the definition of a marking [Rei13] on a distributed data scheme. A marking is the core concept to describe the statics of a distributed data-aware system.

Intuitively, populations of data entities are distributed over locations in a distributed environment. Examples are a database, a mailbox, or a folder, but also states or some logical conditions of entities. These locations may be structured, contain other locations, and may be logically or physically distributed. In our model, we model locations by *places* which can be seen as atomic locations. Here, atomic means that no structure is assumed on places. We use the term "place" to stay in line with the terminology used in Petri net theory. Abstracting locations to places is a simple and straight-forward formalization.

For technical reasons, we assume that the terms stored in one place all have the same sort. Hence, a place can be modeled by a single variable of that sort. A distributed data scheme is a data scheme with a distinct finite set of places.

Definition 8 (Distributed Data Scheme)
Let $(F, \equiv)$ be a data scheme. Let $P \subseteq \mathbb{V}$ be a finite set of variables with $\textsc{Sort}(p) \in \textsc{Sorts}(F)$ for all $p \in P$.
Then, $(P, F, \equiv)$ is a *distributed data scheme* with *places* P.

Now, we recall *markings* of a distributed data scheme to model distributed data. In our formalization, a marking assigns a bag of ground terms to each place. We lift the congruence of bags of ground terms to markings. A formal definition can be found in Section A.5.

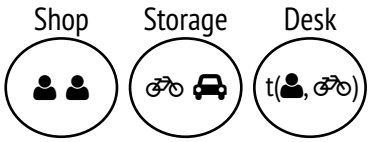
Definition 9 (Marking)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $m \in \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P}$ and for all $p \in P$:

$$\textsc{Sorts}(\text{support}(m_p)) = \{\textsc{Sort}(p)\}$$

Then, m is a *marking*.

Let $m' \in \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P}$ be a marking and for all $p \in P$: $m_p \equiv m'_p$.
Then, m and m' are *equivalent*, written $m \equiv m'$.

In Figure 11 a marking over the three places Shop, Storage, Desk is shown. As usual, we depict locations as ellipses and write the ground terms of the assigned bags into the ellipses.



(a)

place p:	Shop	Storage	Desk
m_p^{11}:	[,]	[,]	[t(,)]

(b)

Figure 11.: The marking m^{11} of the data scheme $\left(F_{10.3}, \equiv_{A_{10.3}}\right)$ from Figure 10a and the three places Shop, Storage, and Desk. m^{11} is visualized using ellipses in (a) and as a table in (b).

PARAMETERIZED EFFECTS. For technical reasons, we introduce the following definition. *Parameterized effects* generalize markings in two ways: First, the polynomial is not restricted to semi-positive values. Accordingly, parameterized effects are closed under subtraction. Second, the terms may contain variables. This way, a parameterized effect may represent an infinite set of non-parameterized effects.

Definition 10 ((Parameterized) Effect)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $V \subseteq \mathbb{V}$. Let $e \in \mathbb{Z}\langle {}^{V}_{F} \rangle^{P}$ and for all $p \in P$:

$$\textsc{Sorts}(\text{support}(e_p)) = \{\textsc{Sort}(p)\}$$

Then, e is a V-*parameterized effect of* $(P, F, \equiv)$.

Let $e' \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P}$ be a parameterized effect and for all $p \in P$: $e_p \equiv e'_p$.
Then, e and e' are *equivalent*, written $e \equiv e'$.

If $V = \emptyset$, we omit "$\emptyset$-parameterized" and simply write "effect". If $V = \mathbb{V}$ or obvious from the context, we write parameterized effect. We observe that every marking is a parameterized effect. Hence, we call a semi-positive V-parameterized effect a V-parameterized marking.

We observe that for a distributed data scheme $(P, F, \equiv)$, the lifted relation $\equiv \subseteq \mathbb{Z}\langle {}^{F}_{\mathbb{V}} \rangle^{P} \times \mathbb{Z}\langle {}^{F}_{\mathbb{V}} \rangle^{P}$ is an equivalence relation.

2.2 ALGEBRAIC PETRI NETS

In this section, we recall the definitions of *transitions* and *algebraic Petri nets* [Rei91]. We define *steps of transitions* based on the *monotone closure* and observe that a set of steps is *monotone, bounded, and structured* if and only if it is induced by a finite set of transitions.

places:	Shop	Storage	Desk
$m^{12.1}$:	[👤, 👤]	[🚲, 🚗]	[ ]
$m^{12.2}$:	[👤]	[🚗]	[tuple(👤,🚲)]
$m^{12.3}$:	[👤]	[🚲]	[ ]
$m^{12.4}$:	[ ]	[ ]	[tuple(👤,🚲)]

Figure 12.: Two steps $\left(m^{12.1}, m^{12.2}\right)$ and $\left(m^{12.3}, m^{12.4}\right)$ over the data scheme $(F_{10.3}, A_{10.3})$ with two places.

In the previous section, we showed how to model the statics of distributed data-aware systems as markings of a distributed data scheme. In this section, we will define a model of dynamics of a distributed data scheme by means of *transitions* over a distributed system. The resulting formalism is known as algebraic Petri nets [Rei91; Rei13].

The dynamics of a distributed system are modeled by a set of steps. A step is a pair of markings. The set of steps of a distributed data-aware system is infinite in general. A transition induces an infinite set of steps and thus model the dynamics of a distributed data-aware system. We define transitions and algebraic Petri nets in Section 2.2.1.

Not every set of steps is induced by a finite set of transitions. Sets of steps, which are not induced by a finite sets of transitions are out of scope of this thesis. In Section 2.2.2, we present a characterization of sets of steps which are induced by a finite set of transitions.

For this section, we fix a distributed data scheme $(P, F, \equiv)$.

STEPS. A distributed data scheme induces the set of markings. A step is a pair of markings, called source and target marking. For technical reasons, we define parameterized steps based on parameterized markings.

> **Definition 11** (Step)
> Let $V \subseteq \mathbb{V}$. Let $(m, m') \in \mathbb{N}\langle {}^{V}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{V}_{F}\rangle^{P}$ be a pair of V-parameterized markings.
>
> Then, (m, m') is a *V-parameterized step of* $(P, F, \equiv)$.
>
> We call m the *source marking of* (m, m'), m' the *target marking of* (m, m'), and $m' - m$ the *effect of* (m, m').

We write just "step", instead of "$\emptyset$-parameterized step".

In Figure 12 the step $\left(m^{12.1}, m^{12.2}\right)$ is shown. Intuitively, one customer in the Shop orders a bicycle from the Storage. Then, a tuple of both objects, tuple(,), is produced on the Desk. The distributed data scheme is $\left(F_{10.3}, \equiv_{A_{10.3}}\right)$ from Figure 10a with the places Shop, Storage, and Desk.

2.2.1 Transitions

In the following, we recall transitions and the steps of a transition. Then, we define algebraic Petri nets and reachability in an algebraic Petri net.

Definition 12 (Transition)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $t = \left(t^-, t^+\right) \in \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^P \times \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^P$.

Then, t is a *transition* over F and P. We refer to t^- as *precondition*, to t^+ as *postcondition*. Moreover, $t^\Delta = t^+ - t^-$ is the *effect of* t.

For figures, we use the established graphical notation for a transition t: A square which is connected to places via arcs. For every place $p \in P$, we depict an arc from p to t with inscription t^-_p. and an arc from t to p with inscription t^+_p. For a bag $[x]$ we write x as arc inscription to avoid notational overhead. As usual, we omit arcs with inscription $[\,]$ emphasizing the locality of the precondition and postcondition.

In the example in Figure 13b, the transition order is shown. Intuitively, the example models any step where a customer orders an object while the next object, modeled by the function n, is produced. The order is stored as a tuple at the location Orders. The precondition is $[\,]$ for Orders, $[\mathbf{c}]$ for Customers, and $[n(\mathbf{o})]$ for Objects. The postcondition is $[t(\mathbf{c}, \mathbf{o})]$ for Orders, $[\,]$ for Customers, and $[n(\mathbf{o})]$ for Objects.

In the following we define the steps of a transition based on the realizations and the monotone closure (Section A.6) of the transition.

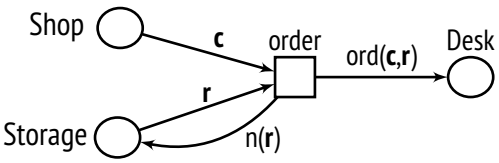
Definition 13 (Steps of a Transition)
Let $\left(t^-, t^+\right) \in \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^P \times \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^P$ be a transition. Let $m \in \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^P$ be a marking. Let $\left(\tau^-, \tau^+\right) \in \mathcal{R}\left(t^-, t^+\right)^\uparrow$.

Then $\left(m + \tau^-, m + \tau^+\right)$ is *step of* t.

In Figure 13c, some example of steps are shown that are induced by the transition order from Figure 13b. Intuitively, order consumes a customer from the Shop and a product from the Storage producing a tuple of both o(,) on the Desk. We observe that for $\sigma \in \{\mathbf{c}, \mathbf{r}\} \overset{F}{\rightsquigarrow} \emptyset$ with $\sigma(\mathbf{c}) =$ and $\sigma(\mathbf{r}) =$, we have that $m^{13.1} = \sigma(\text{order}^-)$ and $m^{13.2} =$

places P_{13}:	Shop, Storage, Desk
function symbols F_{13}:	👤 : C, 📦 : R,
	n : R → R,
	ord : C × R → O
axioms A_{13}:	∅
sorts $\text{SORT}(F_{13})$:	C, R, O
variables:	$\mathbf{c} \in \mathbb{V}_C$, $\mathbf{r} \in \mathbb{V}_R$

(a) The distributed data scheme $(P_{13}, F_{13}, \equiv_{A_{13}})$ with sorts and some variables.



(b) The transition order over $(P_{13}, F_{13}, \equiv_{A_{13}})$.

place:	Shop	Storage	Desk
$m^{13.1}$:	[👤]	[📦]	[ ]
$m^{13.2}$:	[ ]	[n(📦)]	[ord(👤,📦)]
$m^{13.3}$:	[👤]	[ ]	[ ]

(c) Three markings over $(P_{13}, F_{13}, \equiv_{A_{13}})$.

Figure 13.: The transition order with distributed data scheme $(P_{13}, F_{13}, \equiv_{A_{13}})$.

$\sigma(\mathsf{order}^+)$. Accordingly, we deduce that $(m^{13.1}, m^{13.2}) \in \mathcal{R}(\mathsf{order})$ is a step of order. Moreover, an additional, unaffected customer may be in the Shop. Hence, $(m^{13.1} + m^{13.3}, m^{13.2} + m^{13.3})$ is a step of order.

We observe that our definition is equivalent to considering steps induced by transitions *enabled* in *firing modes* as in [Rei13].

Algebraic Petri nets, Reachability

In this section, we first define runs and reachable markings. We complete the formalization with the model of data-aware distributed systems used in this thesis: algebraic Petri nets.

REACHABLE MARKINGS. Starting from a given initial marking, a system may progress in steps consecutively and reach other markings.

Definition 14 (Reachable)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $\Psi \subseteq \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P \times \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$ be a set of steps. Let Ψ^* be the reflexive transitive closure of Ψ. Let $m \in \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$.

Then, $\textsc{Reach}_\Psi(m) = \left\{ m' \in \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P \mid (m, m') \in \Psi^* \right\}$ is the set of Ψ-*reachable markings from* m.

If the set of steps Ψ is obvious from the context we also write *reachable* instead of Ψ-reachable. For a set T of transitions and its induced set of steps S we also write T-reachable instead of S-reachable.

ALGEBRAIC PETRI NET. An algebraic Petri net structure consists of a distributed data scheme and a finite set of transitions. If we consider an algebraic Petri net structure with an initial marking, we have an algebraic Petri net.

Definition 15 (Algebraic Petri Net)
Let $(P, F, \equiv)$ be distributed data scheme, $T \subset \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P \times \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ be a finite set of transitions, and $m^0 \in \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$ be a marking.

Then, $S = (P, F, \equiv, T)$ is an *algebraic Petri net structure* and $\left(S, m^0\right)$ is an *algebraic Petri net*.

The set $\textsc{Reach}_T(m^0)$ is the set of *reachable markings of* $\left(S, m^0\right)$.

In Figure 14, we see an example for an algebraic Petri net. As usual, we draw the initial marking directly into the ellipses of the places. The example has two transitions: order-produce for orders and production of new products in the storage, and delete for deleting orders.

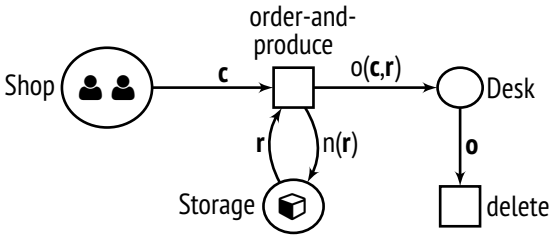


Figure 14.: An algebraic Petri net over the distributed data scheme $(P_{13}, F_{13}, \equiv_{A_{13}})$ from Figure 13a.

2.2.2 Monotone, Bounded, and Structured Set of Steps

In this section, we characterize which sets of steps can be describe by transitions. To this end, we formalize the following properties of a set of steps:

- monotonicity, i.e. source and target marking can be arbitrary enlarged
- bounded causation, i.e. there exists a bound for minimal source markings
- bounded effect, i.e. there exists bound for the changed terms of each step
- structured with respect to term functions, i.e. affected terms can be described finitely

Using this formalization, we prove the following theorem: A set of steps is induced by a finite set of transitions are if and only if the set of steps satisfies the four assumptions.

Theorem 16 (Steps of Transitions are Monotone, Bounded, and Structured Sets of Steps and Vice Versa)
Let $\Psi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P}$ be a set of steps. Then, the following statements are equivalent:

1. Ψ is monotone, precondition bounded, effect bounded and structured with respect to term functions.
2. There exists a finite set of transitions $T \subset \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^{P}$ such that Ψ is the set of all steps of T.

Proof of Theorem 16. Follows directly from applying forthcoming Lemma 23 (Page 52).

In this section, we use the *weight sum of an effect* $e \in \mathbb{Z}\langle {}^{\mathbb{F}}_{\mathbb{V}} \rangle^P$, $\|e\|$, defined by:

$$\|e\| = \sum_{p \in P} \sum_{\theta \in \langle {}^{\mathbb{V}}_{\mathbb{F}} \rangle} e(\theta)$$

MONOTONE SET OF STEPS. In the next definition, we formalize the first property of a set of steps: *monotonicity*. Intuitively, a set of steps Ψ is monotone, if, for each step of Ψ, adding more, entities to the source and target marking, also results in a step of Ψ.

Definition 17 (Monotone, Monotone Closure)
Let $\Psi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P \times \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$ be a set of steps, such that for all $(\psi, \psi') \in \Psi$ and markings $m \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$:

$$(m + \psi, m + \psi') \in \Psi$$

Then, Ψ is *monotone*.

For a set of steps $\Psi' \subseteq \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P \times \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$, the *monotone closure* $\Psi'^{\uparrow}$ is the least monotone set of steps containing Ψ'.

The steps $\left(m^{12.1}, m^{12.2}\right)$, $\left(m^{12.3}, m^{12.4}\right)$ from Figure 12 illustrate monotonicity: In $m^{12.3}$ and $m^{12.4}$, there is an additional customer in Shop and an additional car in Storage, compared to $m^{12.1}$ and $m^{12.2}$, respectively. Other than that, both steps are equal. Hence, we have:

$$\left(m^{12.3}, m^{12.4}\right) \in \left\{\left(m^{12.1}, m^{12.2}\right)\right\}^{\uparrow}$$

CAUSATION BOUNDED SET OF STEPS. The second property of sets of steps of is that the causations of the steps are bounded. The *causation* of a step $(\psi, \psi') \in \Psi$ is the minimal source marking of all steps of Ψ with the same effect as (ψ, ψ'). A set of steps Ψ is causation bounded if there exists a bound $b \in \mathbb{N}$, such that for every step $(\psi, \psi') \in \Psi$, there exists a step $\left(\underline{\psi}, \underline{\psi'}\right) \in \Psi$ starting from a marking with less than b terms, but with the same effect as (ψ, ψ').

Definition 18 (Causation Bounded Set of Steps)
Let $\Psi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P \times \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$. Let $(\psi, \psi') \in \Psi$. Then, $\textsc{Caus}(\psi, \psi')$ is the *causation of* (ψ, ψ') *in* Ψ defined by:

$$\textsc{Caus}(\psi, \psi') = \min\left\{\phi \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P \mid (\phi, \phi') \in \Psi,\ \phi' - \phi = \psi - \psi'\right\}$$

> Let $b \in \mathbb{N}$ such that for all $(\psi, \psi') \in \Psi$:
>
> $$\left\|\textsc{Caus}(\psi, \psi')\right\| \leqslant b$$
>
> Then, Ψ is *causation bounded with* b.

We illustrate Definition 18 using Figure 12. Summing the terms in $m^{12.1}$ we obtain: $\left\|m^{12.1}\right\| = 4$. Hence, the singleton set $\{(m^{12.1}, m^{12.2})\}$ is not causation bounded with $b = 2$. Analogously, we have $\left\|m^{12.3}\right\| = 2$. Hence, the singleton set $\{(m^{12.3}, m^{12.4})\}$ is causation bounded with $b = 2$.

However, we observe that both steps have the same effect:

$$m^{12.2} - m^{12.1} = m^{12.4} - m^{12.3}$$

Hence, if we consider the set $\{(m^{12.1}, m^{12.2}), (m^{12.3}, m^{12.4})\}$ both steps have the same causation:

$$\textsc{Caus}(m^{12.3}, m^{12.4}) = \textsc{Caus}(m^{12.1}, m^{12.2}) = m^{12.1}$$

Thus, we deduce that the set $\{(m^{12.1}, m^{12.2}), (m^{12.3}, m^{12.4})\}$ is causation bounded with $b = 2$.

In Corollary 19, we observe that if a set of steps Ψ is causation bounded its monotone closure $\Psi^{\uparrow}$ is also causation bounded.

> **Corollary 19** (Causation Boundedness is Preserved under Monotone Closure)
> Let $\Psi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P}$, $b \in \mathbb{N}$, such that Ψ is causation bounded with b.
>
> Then, $\Psi^{\uparrow}$ is causation bounded with b.

EFFECT-BOUNDED SETS OF STEPS. The third property of a set of steps is that the effect of every step is bounded. Here, the effect refers to the number of terms that are different in source and target marking. Hence, we consider the absolute value $\left|m' - m\right|$ of an effect $m - m'$ of a step (m, m'). A set is effect-bounded if there exists a bound b such that the effect of each step is smaller than b.

> **Definition 20** (Bounded Effect of a set of Steps)
> Let $\Psi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P}$ be a set of steps, let $b \in \mathbb{N}$, such that for all $(\psi, \psi') \in \Psi$ holds that $\left\|\left|\psi - \psi'\right|\right\| \leqslant b$.
>
> Then, Ψ is *effect-bounded with* b.

In Figure 12, the absolute value of the effect of $\left(m^{12.1}, m^{12.2}\right)$ is 3, as 🧍, and 🚲 are consumed, and t(🧍, 🚲) is produced. Accordingly, the step $\left(m^{12.3}, m^{12.4}\right)$ has the same absolute value of its effect.

We will show in Lemma 23 that each set of steps satisfies all four properties if and only if it can be described by a set of transitions. In the following lemma we characterize the sets of steps which satisfy the first three properties. Intuitively, a set of steps is monotone, causation bounded, and effect bounded, if and only it is the monotone closure of a bounded set of steps.

Lemma 21 (Monotone Bounded Set of Steps as Monotone Closure)
Let $\Psi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P}$. Let $b \in \mathbb{N}$. Then, the following statements are equivalent:

1. Ψ is monotone, causation bounded with b, and effect bounded with b.
2. There exists $\underline{\Psi} \subseteq \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{\emptyset}_{F}\rangle^{P}$ with:
   a) $(\psi, \psi') \in \underline{\Psi}$ implies $\|\psi\| \leqslant b$ and $\|\psi'\| \leqslant 2b$, and
   b) $\Psi = \underline{\Psi}^{\uparrow}$.

Proof of Lemma 21. 1.⇒2.: Let $E = \left\{\psi - \psi' \mid (\psi, \psi') \in \Psi\right\}$ be the set of effects. For every effect e let (ψ_e, ψ'_e) be the step with effect e and minimal source marking. Let $\underline{\Psi} = \{\psi_e, \psi'_e \mid e \in E\}$ be all steps with minimal source marking. It remains to show: (1) $\underline{\Psi}$ is bounded and (2) Ψ is the monotone closure of $\underline{\Psi}$.

1. Now, let $(\psi, \psi') \in \underline{\Psi}$. By construction of $\underline{\Psi}$, every step in $\underline{\Psi}$ is causation bounded. Hence, it holds that $\|\psi\| \leqslant b$. Moreover, every effect $\left\|\left|-\psi + \psi'\right|\right\|$ is bounded. And hence, we have:
$$\|\psi'\| = \|\psi\| + \|-\psi + \psi'\| \leqslant \|\psi\| + \left\|\left|-\psi + \psi'\right|\right\| = 2b$$
2. The monotone closure of $\underline{\Psi}$ is a subset of Ψ, as $\underline{\Psi} \subseteq \Psi$ and Ψ is monotone. On the other hand, $\underline{\Psi}$ contains all steps with minimal source marking for every effect. Hence, the monotone closure of $\underline{\Psi}$ contains all steps of Ψ.

2.⇒1.: We show the three properties of the monotone closure of Ψ:

MONOTONICITY: follows by definition of the monotone closure.

$$\left\{ \left(m, m'\right) \in \langle {}^{\emptyset}_{\mathsf{F}_{6.1}} \rangle \times \langle {}^{\emptyset}_{\mathsf{F}_{6.1}} \rangle \mid \begin{array}{l} m_{\mathsf{Gödel}} = [\theta], m'_{\mathsf{Gödel}} = [\,], \text{ and } \theta \text{ models} \\ \text{a Gödel number of a terminating} \\ \text{Turing machine.} \end{array} \right\}$$

Figure 15.: An unstructured set of steps using the data scheme $\left(\mathsf{F}_{6.1}, \equiv_{\mathsf{A}_{6.1}}\right)$ and the place Gödel.

> **CAUSATION BOUNDED:** follows from the fact that every element in $\underline{\Psi}^{\uparrow} \setminus \underline{\Psi}$ is not minimal, and every element in $\underline{\Psi}$ is bounded.
>
> **EFFECT BOUNDED:** As every step in $\underline{\Psi}$ is bounded, the effect is also bounded. Every step from the monotone closure has the same effect as a step of $\underline{\Psi}$. Hence, $\underline{\Psi}^{\uparrow}$ is also bounded.

We observe that Lemma 21 is independent of the underlying data scheme and the terms of the steps. Indeed, this characterization fits, considering data-unaware Petri nets, as we will discuss in Section 2.3.

STRUCTURED SET OF STEPS. As the fourth property of sets of steps, we assume the steps of a distributed system are structured with respect to the entities: Every step that produces and consumes entities may always be performed on a set of entities, which can be specified in a structured way. For example, the increment operation is executable on any natural number. We formalize this assumption as follows: We consider the partition of a set of steps by the number of terms before and after the step. Then, each partition is represented by a finite set of parameterized steps.

> **Definition 22** (Structured Set of Steps)
> Let $\Psi \in \mathbb{N}\langle {}^{\emptyset}_{\mathsf{F}} \rangle^{\mathsf{P}} \times \mathbb{N}\langle {}^{\emptyset}_{\mathsf{F}} \rangle^{\mathsf{P}}$ be a set of steps. Let for each $b, c \in \mathbb{N}$: $A_{b,c} \subseteq \mathbb{N}\langle {}^{\mathbb{V}}_{\mathsf{F}} \rangle^{\mathsf{P}} \times \mathbb{N}\langle {}^{\mathbb{V}}_{\mathsf{F}} \rangle^{\mathsf{P}}$ such that:
>
> 1. $A_{b,c}$ is finite
> 2. $\mathcal{R}\left(A_{b,c}\right) = \left\{(\psi, \psi) \in \Psi \mid \|\psi\| = b \text{ and } \left\|\psi'\right\| = c\right\}$
>
> Then, Ψ is *structured*.

We observe that every finite set of steps is structured. For example in Figure 12, $\{(m^{12.1}, m^{12.2}), (m^{12.3}, m^{12.4})\}$ is structured. As a counterexample, a non-structured set of steps is sketched in Figure 15: Consider the data scheme $\left(\mathsf{F}_{6.1}, \equiv_{\mathsf{A}_{6.1}}\right)$ from Figure 6 modeling the natural numbers with the single location Gödel. The set of steps includes all

steps that consume a natural number that refers to a Gödel number of a halting Turing machine and replaces it with a zero. This set of steps is unstructured, as it can not be represented by a finite set of parameterized steps.

Extending Lemma 21, we deduce that each set of steps fulfills all four assumptions if and only if it is the monotone closure of the set of realizations of a finite set of parameterized steps.

Lemma 23 (Monotone, Bounded, and Structured Set of Steps)
Let $\Psi \subseteq \mathbb{N}\langle {\emptyset \atop F} \rangle^P \times \mathbb{N}\langle {\emptyset \atop F} \rangle^P$. Then, the following statements are equivalent:

1. Ψ is monotone, causation bounded, effect bounded, and structured.
2. There exists a finite set of parameterized steps $\Omega \subset \mathbb{N}\langle {\mathbb{V} \atop F} \rangle^P \times \mathbb{N}\langle {\mathbb{V} \atop F} \rangle^P$ such that $\Psi = \mathcal{R}(\Omega)^{\uparrow}$.

Proof of Lemma 23. 1.⇒2.: By Lemma 21, there exists a set of steps $\underline{\Psi} \subseteq \Psi$, where $\underline{\Psi}$ is bounded and Ψ is the monotone closure of $\underline{\Psi}$. Let $B = \left\{ \|\psi\| \mid (\psi, \psi') \in \underline{\Psi} \right\}$. Let $C = \left\{ \|\psi\| \mid (\psi, \psi') \in \underline{\Psi} \right\}$. Then $B \times C$ is finite. As Ψ is structured, for every (b, c) exists a finite representation $A_{b,c}$ of $\left\{ (\psi, \psi') \in \Psi \mid \|b\| = \psi \text{ and } \psi' = c \right\}$. We consider the union of all:

$$\Omega = \bigcup_{(b,c) \in B \times C} A_{b,c}$$

As $B \times C$ is finite and reach representation is finite, this set is finite. Moreover, by definition $\mathcal{R}(\Omega) \subseteq \Psi$. Additionally, we have $\underline{\Psi} \subseteq \mathcal{R}(\Omega)$. By Lemma 21, we have $\underline{\Psi}^{\uparrow} = \Psi$. And, by definition, we have $\underline{\Psi} \subseteq \mathcal{R}(\Omega)$. Hence, $\underline{\Psi}^{\uparrow} \subseteq \mathcal{R}(\Omega)^{\uparrow}$ and $\Psi \subseteq \mathcal{R}(\Omega)^{\uparrow}$. On the other hand, $\mathcal{R}(\Omega) \subseteq \Psi$ and as Ψ is monotone, also $\mathcal{R}(\Omega)^{\uparrow} \subseteq \Psi$. And we conclude $\mathcal{R}(\Omega)^{\uparrow} = \Psi$.

2.⇒1.: The first three properties follow immediately from applying Lemma 21. It remains to show that $\mathcal{R}(\Omega)^{\uparrow}$ is structured. To this end, let $b, c \in \mathbb{N}$ and

$$A_{b,c} = \left\{ (\psi, \psi') \in \mathcal{R}(\Omega)^{\uparrow} \mid \|\psi\| = b \text{ and } \left\|\psi'\right\| = c \right\} .$$

We observe that every step $(\psi, \psi') \in A_{b,c}$ is of the form

$$(o + m, o' + m),$$

where $(o, o) \in \mathcal{R}\left((\omega, \omega')\right)$ for some $(\omega, \omega) \in \Omega$ and $m \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P}$. Then, for every $m' \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P}$ with $\|m\| = \|m\|$ is also $(\omega + m, \omega' + m) \in A_{b,c}$. Hence, all such m can be represented by replacing every term by a variable resulting in an abstract marking $a_{b,c,\omega} \in \mathbb{N}\langle {}^{V}_{F} \rangle^{P}$ such that: $(o + m, o' + m) \in A_{b,c}$ if and only if $m \in \mathcal{R}(a_{b,c\omega})$. As such $a_{b,c,\omega}$ exists for every ω and Ω is finite, we conclude that $A_{b,c}$ is finitely representable as:

$$A_{b,c} = \mathcal{R}\left(\left\{(\omega + a_{b,c,\omega}, \omega' + a_{b,c,\omega}) \mid \omega \in \Omega\right\}\right).$$

Considering a set of steps that satisfies the four properties, we can specify it by parameterized minimal steps. Each minimal parameterized step corresponds to a transition of an algebraic Petri net.

2.3 DISCUSSION

In Sections 2.1 and 2.2, we recalled algebraic Petri nets as a model for distributed data-aware systems. In this section we discuss each part of the formalization separately in the context of related work. We start with distributed data schemes and markings in Section 2.3.1. We continue with the model of dynamics by means of steps of transitions in Section 2.3.2.

2.3.1 Discussion - Distributed Data Schemes and Markings

In this section, we discuss our model for entities, locations and populations: terms of a distributed data scheme, places, and markings.

DATA SCHEMES AND TERMS. Data schemes provide a way to model entities as terms of function symbols equipped with a congruence. Specifying entities by terms and interpreting them was established in the area of *algebraic specification*. The idea to describe entities with operations by algebraic specification was first discussed in [Gut76; GHM76].

In our approach we restrict ourselves to the Herbrand structure over function symbols, equipped with a congruence. In contrast to that, algebraic specification generally considers any interpretation [ST12]. Moreover, these interpretations may contain elements, which are not described by terms. This are referred to as *junk* [ST12]. In our approach it is not possible to consider models with junk. However,

our approach is equivalent to considering junk-free interpretations: As the interpretation of two terms may be equal, every interpretation induces a congruence on the Herbrand structure. On the other hand, given a data scheme $(F, \equiv)$, there exists a unique (up to isomorphism) junk-free interpretation of F such that two terms of the interpretation are equal if they are equivalent. This interpretation is known as the quotient algebra of $\equiv$. The notations we use in this thesis emphasize the algebraic aspect and the handling of terms, which we exploit throughout the following chapters.

In the 1990s, the *common algebraic specification language* (CASL) was published with the goal to become a standard language for algebraic specification [BM04]. CASL is supported by a tool and models data entities and operations as interpretations of signatures. The notation of a data scheme can be seen as a special case of a CASL model. The signature translates into a set of function symbols. In contrast to CASL, data schemes model predicates as Boolean functions and not explicitly. If the congruence of a data scheme is specified by axioms, the modeling of data schemes is equivalent to usage of the *free* statement in a CASL model. Intuitively, considering exactly the smallest induced congruence of axioms, it is equivalent to avoid *confusion* and *junk*. Moreover, the axioms are negation-free equations, which is a subset of the first-order formulas as it is possible in CASL.

BAGS OF GROUND TERMS. Typically, there are two ways to model a population of data entities: First, using *set semantics*, and second using *bag semantics*. Whereas classical theoretical foundations are build on set semantics, implementations are often built on bag semantics. However, it is well-known that for data-aware systems the difference has a huge impact: For instance in relational algebra, the problem of *conjunctive query containment* is NP-complete under set semantics [CM77], but its exact complexity under bag semantics remains an open problem [CV93; JKV06]. In *datalog*, some query problems are undecidable, which are decidable with set semantics [Fro17]. However, for distributed systems, modeling with set semantics is counter-intuitive: Uniqueness is a global property that cannot be achieved locally. Uniqueness of entities enforces that two otherwise concurrent steps may interfere by duplicates. In a distributed system, duplicates may be desired. Hence, for distributed systems it is common to use bag semantics, as in Petri nets [Rei98; Rei13]. In [MR16a] a combination is used: Whereas the process is modeled distributed, the underlying data is stored in a global database using set semantics. However, in this theses we focus on systems with distributed data.

DATA SCHEMES VS. RELATIONAL ALGEBRA. For the modeling of management and processing of data, *relational algebra* [Cod70] became a well-established theory in the last decades [AHV95]. The idea to integrate those approaches has been discussed in [DMW82], suggesting that algebraic specification is a more general approach to model data entities. The advantage of this generality is that different data models may also be taken into account. For example under the phrase *NoSql*, several technique for e.g. graphs, documents, etc. have been established [Cat10; Cuz+13].

UNIFICATION. It is known since [Rob65] that the unification problem is solvable if the set of axioms is empty. However, for arbitrary sets of axioms, it is known, that this problem is undecidable: The proof is sketched in [BS94]. Intuitively, the 10th Hilbert Problem is reducible for a set of axioms inducing integers with multiplication.

However, there exist classes of term equivalences, where this problem is decidable. In the area of unification theory classes of specifications are studied, where unification problems are decidable and finitely representable. A well-known example the class of associative-commutative and idempotent functions as studied in [Nar96; KN92]. An overview and classification can be found in [BS94]. The proof of Lemma 7 (Page 38) stands in line with related work on many-sorted specifications [Erb+14; BS96; BS92], where disjoint axioms are considered. The proof could also be derived from [Sch86; Sch89], where disjoint axioms are considered. There, the unifiers are combined in a more general setting.

DIS-UNIFICATION. Considering dis-unification problems over commutative, associative and idempotent function symbols and a unit element, the problem is NP-hard [Nar96]. [HK97] study the complexity and study variants where the problem is solvable in polynomial time. [BBM16] study the related case of dis-unification in description logic. More recent work studies unification and dis-unification problems with respect to term rewriting systems [RGN17]. A survey of dis-unification results can be found in [Com91].

Summarizing, an important class of compact data schemes where each unification problem can be represented finitely and for each dis-unification problem it is decidable if it dis-unifiable subsume the following congruence specifications: associative, commutative and idempotent function symbols with unity elements as studied in [Nar96].

MARKINGS MODELING DISTRIBUTED DATA. In Petri net theory, a marking describes a state of a distributed data-aware system [Rei13].

Places are fixed for a given Petri net and do not change during an execution. However, it is possible interpret a Petri net place with different terms as different places in the sense of [Rei98]. Hence, an algebraic Petri net can be seen as a Petri net with an infinite set of places. This enables modeling of dynamic system, where the underlying structure is not fixed, but may change over time. This approach is discussed using the example of an *echo algorithm* in [Kin+97].

There exist different models that distribute data entities in a network, most known under the term distributed database, an overview can be found in [ÖV11].

2.3.2 Discussion - Transitions in Algebraic Petri Nets

Transitions in algebraic Petri nets model the dynamics of distributed data-aware system. Hereby, every transition induces a set of steps, where each step is a pair of markings.

ALGEBRAIC PETRI NETS. In this thesis, we always consider the set of all steps, including unreachable. This yields the possibility to model and reason about the system. Algebraic Petri nets are exactly the model which satisfy the four properties: monotonicity, causation boundedness, effect boundedness and structure. In the literature these properties are mostly written implicitly and not studied explicitly. Carl-Adam Petri tried to motivate his work with assumptions about the universe as in [Pet62; Pet86] using formal reasoning. His dedicated approach was very sophisticated with a fundamental discussion about the laws of physics making him a pioneer of distributed computing. In this thesis, it is not the goal to explain the whole theory of distributed system. However, we strive for the goal to at least characterize the properties we assume about a distributed data-aware system. A similar characterization has been pursued in [Wol14]: There, Petri nets with indistinguishable tokens are considered. The properties of steps are linearity and monotonicity. Here, "linearity" can be seen as equivalent to boundedness: Every step is reducible, as it causation and effect are bounded. As in this thesis, in [Wol14], these property are not restricted to reachable steps.

In the late eighties, several approaches for combing petri nets and algebraic specification have been introduced, e.g. [KS87; BCM87; Vau86]. A brief overview and summary is given in [Rei91] presenting the core techniques. Later, in [Rei13] the same approach is still used and established with many case studies. Most of our definitions follow these approach up to technical details. However, our formalism

of algebraic Petri nets is slightly different with respect to the following three points:

- We restrict ourselves to the Herbrand structure with a congruence. We do not consider arbitrary interpretations of algebraic specification.
- We omit transition guards as it is often done in high-level Petri nets, e.g. [Rei13; EH16]. It is left for future work, how the notation of guards improve the modeling and analysis of data integrity in distributed data-aware systems.
- Our definition of reachability neglects locality of transitions: Typically, distributed runs and unfoldings are a powerful tool for analysis of Petri nets emphasizing its distributed nature [EH08]. In this thesis, we never exploit concurrency and avoid the technical overhead.

In Petri net theory, the property of monotonicity of steps with respect to an order on states led to the theory of *well-structured transition systems* [Fin87; FS01]. However, algebraic Petri nets fall not into the category of well-structured transition system, although the steps are monotone. Intuitively, the manipulation of data objects by terms is not monotone. It is left to future work, whether a relevant class of algebraic Petri nets fall into class of well-structured transition systems.

PROCESSES AND DATA. The presented approach combines modeling techniques from both, process modeling and data modeling. In the last years, this line of research gained more attention, especially in the context of business process modeling. The best known approach for business process modeling is BPMN. However, data modeling is restricted in BPMN and BPMN has only partially formal semantics [Woh+06].

In [Mey+13; MSW11], the authors suggest data-aware extensions of BPMN with formal semantics that support relational models. It is an open question how the approach relates to data-aware approaches such as algebraic Petri nets or colored Petri nets. In [MSW11], the authors classify possible extension of business process model with data. In their categories, algebraic Petri nets are *control and data-driven*.

Other known formal modeling languages are UML Activity Diagrams [Woh+05], BPEL [WKL14] and YAWL [Wyn+09]. All of them have formal semantics and provide analysis techniques. However, they focus around the control flow and abstract from data [LS07].

In the following, we study several approaches with formal semantics:

MONOTONIC DATA-AWARE BUSINESS PROCESSES [MNS14; SMN16] are finite-state transition systems equipped with a database. The data modeling language *Datalog* [Dan+97] models the underlying relational database with respective queries. However, data-aware business processes consider a monolithic process. They do not incorporate distributed systems or communication between different agents.

DATA-CENTRIC DYNAMIC SYSTEMS [Bag+13] combine data and process modeling following a similar approach as data-aware business processes. Transitions are defined based on action rules and database queries. In contrast to data-aware business processes, the state space is defined implicitly by all states of the database which may yield an infinite transition system. The formalism enables modeling remote service calls by uninterpreted functions. However, communication between agents is not considered. Hence, modeling of distributed system is only partially possible.

ARTIFACT CENTRIC BUSINESS MODELS [Hul08] can be considered as a general framework for combinations of processes and data. One formal model is that of [Fah+11]. The approach focuses on the life-cycle of a data object and does not treat them as resources of behavior. In the recent years, the Guard-Stage-Milestone model has been suggested for artifact-centric modeling [DHV13; Hul+10; PD12]. The approach focuses on a single artifact. Thus, modeling of distributed systems is not immediately possible. One approach is described in [Ali+18], where distributed aspects of a business process are combined with artifact-centric modeling.

DATA DEPENDENT SERVICES [BG14; Wag15] model services that operate with using algebraic Petri nets with interfaces. Here, only partial knowledge about the system is assumed. For the specification, termination and temporal specifications are considered.

PETRI NETS WITH DATA [Las16] consider Petri nets with distinguishable tokens. In [Las16] different variants are discussed. The *Well-Quasi-Order dichotomy* is conjectured for analysis. However, algebraic Petri nets fall on the "difficult side" of that dichotomy.

PETRI NETS WITH NAMES [RF10] model data by pure names. The model is in the class of well-structured transition systems. Here, no structure on the data is assumed, which is one of our assumption of distributed data-aware systems. An extension also

considers vectors of pure names, which can model tuples of relational databases specifically for the analysis of data-aware systems [Hee+09; Wer11].

COLOURED PETRI NETS: Coloured Petri nets are an established tool to model and simulate distributed data-aware systems. The language used to described is very expressive, such that the set of steps can be specified in more complex way [JK09; JK15]. However, the systems describe are more complex than those induced by the properties we postulate in this thesis.

DB-NETS [MR16a] are an expressive modeling language focusing on the modeling of processes sharing a global database. Similar as in our approach, they put a focus on tuples of objects from a domain. However, in DB-Nets, it is possible to express unbounded causation and unbounded effects. Data manipulations of database is possible with an expressive query language. In their model, the authors assume a globally accessible database. However, in this thesis we focus on systems with distributed data.

3 DATA INTEGRITY IN DISTRIBUTED SYSTEMS

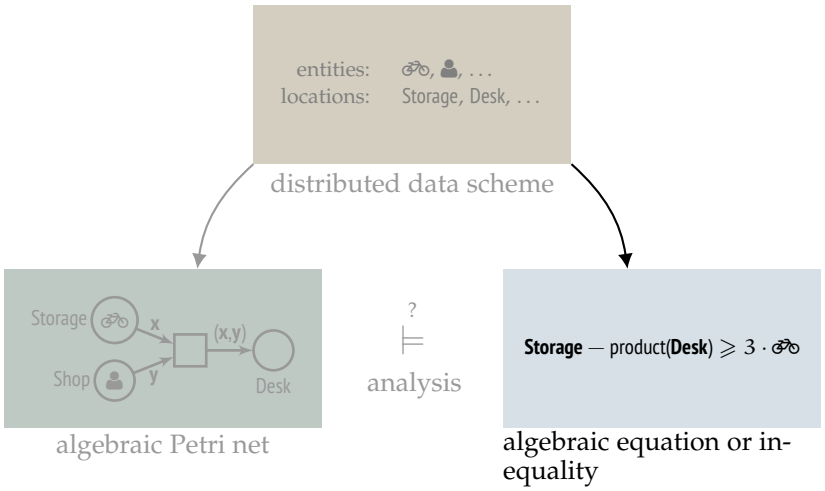


Figure 16.: Overview of part II. The contents of chapter 2 and 4 are grayed out.

In the previous chapter, we studied algebraic Petri nets as a formal model for distributed data-aware systems. In this chapter, we explore how *data integrity* can be specified in an algebraic Petri net. We rely on the known concept of an *algebraic equation* or *algebraic inequality* [Vau85; Rei13]. As indicated in Figure 16, an algebraic equation or inequality depends only on the distributed data scheme and not on the algebraic Petri net. We first observe that data integrity is a state property in Section 3.1. Then, we define *abstraction queries* built of *functions induced by terms* and *multiplication* in Section 3.2. Based thereon, we recall algebraic equations and inequalities in Section 3.3.

The term "data integrity" is overloaded in computer science. Data integrity requirements captured by algebraic equations and inequalities are specific to distributed data schemes and are different from notions used in classical databases [AHV95]. We end the chapter with a discussion of related work in Section 3.4.

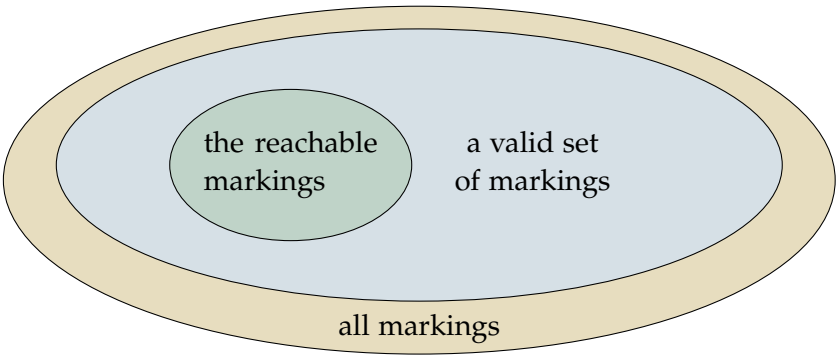


Figure 17.: Validity shown as Venn diagram.

3.1 DATA INTEGRITY

The term "data integrity" is heavily overloaded, since it appears in various areas of computer science with different meanings. In this thesis, we refer to data integrity as the logical integrity between entities of distributed data. Logical integrity is a property, which depends on the semantics of the data. A typical example for data integrity is *referential integrity* among referential data, or *mutual exclusion* of critical resources. Data integrity is a semantic property of the data that discriminates between valid and invalid markings.

Formally, we can define data integrity as a subset of markings of a distributed data scheme. Then, a subset of markings ϕ is valid in an algebraic Petri net, if every reachable marking is in ϕ.

Definition 24 (Valid)
Let $\left(P, F, \equiv, T, m^0\right)$ be an algebraic Petri net. Let $\phi \subseteq \mathbb{N}\langle{}^{\emptyset}_{F}\rangle^P$. Let $m \in \mathbb{N}\langle{}^{\emptyset}_{F}\rangle^P$.

Then, ϕ is *valid* in m if $m \in \phi$, and ϕ is *valid* in $\left(P, F, \equiv, T, m^0\right)$ if $\textsc{Reach}_T(m^0) \subseteq \phi$.

Validity is shown as a Venn diagram in Figure 17: Here, the blue set of markings is a valid set of markings as it is a superset of the reachable markings.

In the following sections, we discuss how data integrity can be specified by means of algebraic equations and inequalities, where each is a finite representation of an infinite set of markings. An algebraic equation or inequality contains a left-hand side and a right-hand side. Whereas the right-hand side is a constant, the left-hand side is described by an *abstraction query*. Equivalently, in linear algebra, an equation or inequality contains a linear polynomial as the left-hand side and a constant as the right-hand side.

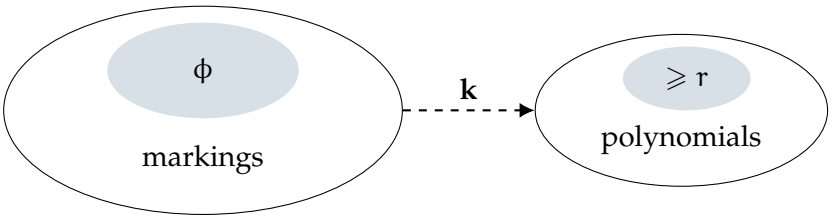


Figure 18.: Integrity constraint ϕ described by an abstraction query $\mathbf{k}$ and a constant r.

The concept of algebraic equations and inequalities is sketched in Figure 18: The set of markings is depicted as the left ellipse. An abstraction query $\mathbf{k}$ is a function that maps each marking to a polynomial. A polynomial is generalization of a bag, that assigns integers to terms. The polynomials are depicted as the right ellipse. There, the polynomials greater or equal r are depicted in blue. Accordingly, all markings m with $\mathbf{k}(m) \geqslant r$ are depicted as the blue set, ϕ, on left side. The set of markings ϕ is hence described by the abstraction query $\mathbf{k}$ and the polynomial r.

Algebraic inequalities over bag semantics correspond to *inclusion dependencies* [AHV95] over set semantics. Here, the set inclusion $\subseteq$ corresponds to the partial order $\leqslant$. We discuss this relationship in more in detail in Section 3.4.

3.2 ABSTRACTION QUERIES

In this section, we study *abstraction queries*, which map markings to term polynomials using a *term induced function* and an *integer coefficient* for each place.

Based on both classes of functions on effects, we define abstraction queries.

We fix a distributed data scheme $(P, F, \equiv)$ for this section. For technical reasons, we define all functions for parameterized effects, which subsume markings. We will exploit this more general definition in Part II.

3.2.1 Term-induced Functions

A set $F \subseteq \mathbb{F}$ of function symbols induces a set of functions on terms resulting in the Herbrand structure. Moreover, every term using one variable induces a unary function on terms: Intuitively, we replace the variable of the term by another term. For terms κ, θ with $\mathbb{V}(\kappa) = \{x\}$

function symbols F_{20}		axioms A_{20}	terms $\langle^{\emptyset}_{\mathsf{F}_{20}}\rangle$
0 :	$\to N$	projection(tuple(**x**,**y**)) $\doteq$ **x**	0, s(0), s(s(0)),
s :	$N \to N$		📄_0, $\text{📄}_{s(0)}$, …
📄 :	$N \to A$		🗀_0, $\text{🗀}_{s(0)}$, …
🗀 :	$N \to O$		tuple(🗀_0, 📄_0), …
tuple :	$A \times O \to U$		
projection :	$U \to A$		

Figure 19.: The data scheme $(\mathsf{F}_{20}, \equiv_{\mathsf{A}_{20}})$ for files, folders, and tuples, and projection of them with variables **x**,**y** of sort A and O, respectively.

and $\textsc{Sort}(x) = \textsc{Sort}(\theta)$, we denote the term $\sigma(\kappa)$, where $\sigma(x) = \theta$, by $\kappa(\theta)$. We lift a function induced by a term to polynomials of terms. We consider terms using one variable and effects over terms of one sort.

Definition 25 (Term-Induced Functions on Polynomials)
Let $s \in \textsc{Sorts}(F)$, $x \in \mathbb{V}$ with $\textsc{Sort}(x) = s$. Let $r \in \mathbb{Z}\langle^{\mathbb{V}}_{F}\rangle$ be a polynomial with $\textsc{Sort}(\rho) = s$ for all $\rho \in \text{support}(r)$. Let $\kappa \in \langle^{\{x\}}_{F}\rangle$. Then, we define for $\theta \in \langle^{\mathbb{V}}_{F}\rangle$:

$$\big(\kappa(r)\big)(\theta) = \sum_{\substack{\theta' \in \text{support}(r) \\ \kappa(\theta') = \theta}} r(\theta')$$

Figure 20 shows an example. Intuitively, some files belong to certain folders, which is modeled by tuples. The purpose of this example is to show that we can use data operations on polynomials to implement a projection function. The data scheme $(\mathsf{F}_{20}, \equiv_{\mathsf{A}_{20}})$ in Figure 19 induces indices 0, s(0), …, files 📄_i, folders 🗀_i and tuples of both such as tuple(📄_i,🗀_j) for indices i, j. Furthermore, the data scheme uses the function symbol projection. The according axiom induces the common semantics: The result is equivalent to the folder of each tuple. In Figure 20a the bag B of ground terms is shown. In the second column, the term projection(**z**) with the variable **z** of sort U is shown, which induces a function on polynomials of terms. The resulting bag $\kappa(B)$ is shown in the right column. The application is done purely syntactically. However, we observe the equivalence shown in Figure 20b, which fits the intuitive semantics.

bag of terms B	term κ	$\kappa(B)$
[tuple(🗀_0, 🗎_0), tuple(🗀_0, $\text{🗎}_{s(0)}$), tuple($\text{🗀}_{s(0)}$, $\text{🗎}_{s(0)}$)]	projection(**z**)	[projection(tuple(🗀_0, 🗎_0)), projection(tuple(🗀_0, $\text{🗎}_{s(0)}$)), projection(tuple($\text{🗀}_{s(0)}$, 🗎_0))]

(a) An application of a term-induced function term using variable **z** with sort *U*.

[project(tuple(🗀_0, 🗎_0)),
projection(tuple(🗀_0, $\text{🗎}_{s(0)}$)),
projection(tuple($\text{🗀}_{s(0)}$, 🗎_0))] $\equiv_{A_{20}}$ [🗀_0, 🗀_0, $\text{🗀}_{s(0)}$]

(b) The equivalence $\equiv_{A_{20}}$ on $\mathbb{N}\langle {}^{\emptyset}_{F_{20}} \rangle \times \mathbb{N}\langle {}^{\emptyset}_{F_{20}} \rangle$.

Figure 20.: The term-induced function of projection(**x**) applied to a bag of terms.

3.2.2 Multiplication

We can manipulate the assigned integers of an effect by multiplying its values using integer multiplication.

As polynomials are an Abelian group, they induce a unique integer module. Accordingly, multiplication is defined in the following way:

Definition 26 (Polynomial Multiplication)
Let $\lambda \in \mathbb{Z}$ be an integer. Let $r \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle$ be a polynomial.

Then, $\lambda r \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle$ is the *product of* r *and* λ defined for each $\theta \in \langle {}^{\mathbb{V}}_{F} \rangle$ by:

$$(\lambda r)(\theta) = \lambda\left(r(\theta)\right)$$

We lift this notion to effects. Let $e \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle^P$ and a vector of integers $\ell \in \mathbb{Z}^P$. Then, the *product of* e *and* λ is defined by:

$$(\ell e)_p = \ell_p e_p$$

Figure 21 shows an example. The data scheme $(F_{21}, \equiv_\emptyset)$ in Figure 21a induces terms that model natural numbers. The bag B shown in Figure 21b contains three terms and the factor is 2. Accordingly, the resulting bag λB contains six terms, as shown in the right column. Although the entities that are modeled can be interpreted as natural numbers, the multiplication is independent of the modeled value.

function symbols F_{21}		axioms	terms $\langle {}^{\mathsf{F}_{21}}_{\emptyset} \rangle$
0 :	$\to N$	-	0, s(0), s(s(0)), ...
s :	$N \to N$		

(a) The data scheme $(\mathsf{F}_{21}, \equiv_{\emptyset})$ for natural numbers.

bag B	factor λ	λB
[0, 0, s(s(0))]	2	[0, 0, 0, 0, s(s(0)), s(s(0))]

(b) The result of a bag of terms multiplied with 2.

Figure 21.: An example of multiplication of a bag of terms

The definition is not restricted to semi-positive polynomials and positive integers. Respectively, one may also apply multiplication of negative integers, for example to express the difference between polynomials.

3.2.3 Abstraction Queries

In this section, we introduce abstraction queries. An abstraction query is made of a P-vector of terms over one variable, and a P-vector of integer coefficients. Each abstraction query induces a function from effects to polynomials.

Definition 27 (Abstraction Query)
Let $\boldsymbol{\ell} \in \mathbb{Z}^P$. Let $\mathbf{o} \in \langle {}^{P}_{F} \rangle^P$ with $\mathbb{V}(\mathbf{o}_p) = \{p\}$ for all $p \in P$.
Then, $(\boldsymbol{\ell}, \mathbf{o})$ is an *abstraction query*.
Let $e \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle^P$ be an effect. Then, $(\boldsymbol{\ell}, \mathbf{o}) \odot e$ denotes the *application of* $(\boldsymbol{\ell}, \mathbf{o})$ *to* e, defined by:

$$(\boldsymbol{\ell}, \mathbf{o}) \odot e = \sum_{p \in P} \ell_p \left(\mathbf{o}_p \left(e_p \right) \right)$$

In Figure 22 an example of an abstraction query and its application is shown. The used data scheme $(\mathsf{F}_{22}, \equiv_{\mathsf{A}_{22}})$ is shown in Figure 22a. It models indices by the sort *Ind*, products over indexes by sort *Pr*, payment by sort *Pay*, and orders by sort *Ord*. The function symbol 🚲 assigns indices to products. Intuitively, that is a bicycle with a product number. The model uses a singleton to model payment. Orders are modeled using the tuple function symbol ord. An order is a tuple of a product and payment. The example models two places: Storage

function symbols F_{22}		axioms A_{22}	terms $\langle \emptyset_{\mathsf{F}_{22}} \rangle$
0 :	$\to$ *Ind*	$\mathsf{prod}(\mathsf{ord}(\mathbf{x},\mathbf{y})) \doteq \mathbf{x}$	0, s(0), s(s(0)), 🏦,
s :	*Ind* $\to$ *Ind*		ord(🚲$_0$, 🏦),
🚲 :	*Ind* $\to$ *Prod*		ord(🚲$_{\mathsf{s}(0)}$, 🏦), ...
🏦 :	$\to$ *Pay*		
ord :	*Prod* $\times$ *Pay* $\to$ *Ord*		
prod :	*Ord* $\to$ *Prod*		

(a) The data scheme $(\mathsf{F}_{22}, \equiv_{\mathsf{A}_{22}})$ modeling indices (sort *Ind*), products (sort *Prod*), payment (sort *Pay*), and order (sort *Ord*) using variables **x** and **y** of sort *Prod* and *Pay*, respectively.

place $\mathrm{p} \in \mathsf{P}_{22}$:	Storage	Shop
term $\mathbf{o}_{\mathrm{p}}$:	**Storage**	prod(**Shop**)
coefficient ℓ_{p}:	1	-1

(b) An abstraction query $\mathbf{k} = (\ell, \mathbf{o})$ over $(\mathsf{P}_{22}, \mathsf{F}_{22}, \equiv_{\mathsf{A}_{22}})$

i	$\mathsf{m}^{22.\mathrm{i}}_{\mathsf{Storage}}$	$\mathsf{m}^{22.\mathrm{i}}_{\mathsf{Shop}}$	$\mathbf{k} \odot \mathsf{m}^{22.\mathrm{i}}$
1	[🚲$_0$, 🚲$_{\mathsf{s}(0)}$]	[ord(🚲$_0$, 🏦)]	[🚲$_{\mathsf{s}(0)}$]
2	[🚲$_0$, 🚲$_{\mathsf{s}(0)}$]	[ord(🚲$_0$, 🏦), ord(🚲$_{\mathsf{s}(\mathsf{s}(0))}$, 🏦)]	[🚲$_{\mathsf{s}(0)}$] $-$ [🚲$_{\mathsf{s}(\mathsf{s}(0))}$]

(c) Two markings $\mathsf{m}^{22.1}$ and $\mathsf{m}^{22.2}$ and $\mathbf{k}$ applied to them.

$$\mathbf{k} \odot \mathsf{P}_{22} \mathrel{\dot{\geqq}}_{\mathsf{A}_{22}} [\,]$$

(d) The algebraic inequality I_{22}.

$$1 \cdot \mathbf{Storage} - 1 \cdot \mathsf{prod}(\mathbf{Shop}) \mathrel{\dot{\geqq}}_{\mathsf{A}_{22}} [\,]$$

(e) Alternative notation of I_{22}.

Figure 22.: An example of an algebraic inequality using an abstraction query.

and Shop. Orders can be stored in the Shop and products can be stored in the Storage.

In this example, data integrity is defined with respect to orders of products: It is required that there exists a distinct product in the storage for every order in the shop. To formalize this, the abstraction does the following: Request all stored products and subtract all ordered products. Hence, the query returns all remaining products. If the result is negative, there are ordered products that are not in the storage.

In Figure 22b, the query is formalized as abstraction query $\mathbf{k}$. It consists of a term $\mathbf{o}_p$ and an integer coefficient ℓ_p for each place p. For the place Storage, the term $\mathbf{o}_{\mathsf{Storage}}$ is Storage: The induced function on polynomials by this term is the identity. For the place Shop, the prod function symbol is applied. By the congruence $\equiv_{A_{22}}$, this is equivalent to projecting to the product of an order. Accordingly, for the place Shop, only the product of the order is considered. Finally, the result of Shop is multiplied by -1. This way, the result contains positive values for all products in the storage and subtract all products which are ordered.

In Figure 22c the markings $m^{22.1}$ and $m^{22.1}$ are shown together with the result when applying abstraction query $\mathbf{k}$. Regarding $m^{22.1}$, for Storage, the result is $[🚲_0, 🚲_{s(0)}]$, for Shop, the result is $[\mathsf{prod}(\mathsf{ord}(🚲_0, 🔑))]$. Accordingly, the sum yields:

$$\left[🚲_0, 🚲_{s(0)}\right] - \left[\mathsf{prod}(\mathsf{ord}(🚲_0, 🔑))\right] \equiv_{A_{22}} \left[🚲_0, 🚲_{s(0)}\right] - \left[🚲_0\right] = \left[🚲_{s(0)}\right] .$$

Their sum is the bag containing $🚲_{s(0)}$ once, intuitively indicating, that there exists the product $🚲_{s(0)}$ that is not ordered.

On the contrary, for $m^{22.2}$ the result is equivalent to the difference $\left[🚲_{s(0)}\right] - \left[🚲_{s(s(0))}\right]$. Here, again the product $🚲_{s(0)}$ is not ordered, but in the storage. However, the product $🚲_{s(s(0))}$ is ordered, but not in the storage. We observe that there are ordered products without the product in the storage if and only if the abstraction query does not return only positive values. We will use this observation in the next section to define integrity constraints build on abstraction queries.

We observe an important property of abstraction queries: They distribute over addition of effects:

Lemma 28 (Additivity of Abstraction Queries)
Let $\mathbf{k}$ be an abstraction query.
Then, the following identity holds for all $e, e \in \mathbb{Z}\langle {}^{\mathbb{V}}_{\mathsf{F}} \rangle^{\mathrm{P}}$:

$$\mathbf{k} \odot \left(e + e'\right) = \mathbf{k} \odot e + \mathbf{k} \odot e'$$

Proof of Lemma 28. Let $(\mathbf{k} = \ell, \mathbf{o})$. Then, we have:

$$
\begin{aligned}
(\ell, \mathbf{o}) \odot (e + e') &= \sum_{p \in P} \ell_p \left(\mathbf{o}_p \left(e_p + e'_p \right) \right) \\
&= \sum_{p \in P} \ell_p \left(\sum_{\substack{\theta' \in \langle \frac{\mathbb{V}}{F} \rangle \\ \mathbf{o}_p(\theta') = \theta}} e(\theta') + e'(\theta') \right) \\
&= \sum_{p \in P} \ell_p \left(\sum_{\substack{\theta' \in \langle \frac{\mathbb{V}}{F} \rangle \\ \mathbf{o}_p(\theta') = \theta}} e(\theta') + \sum_{\substack{\theta' \in \langle \frac{\mathbb{V}}{F} \rangle \\ \mathbf{o}_p(\theta') = \theta}} e'(\theta') \right) \\
&= \sum_{p \in P} \ell_p \left(\mathbf{o}_p \left(e_p \right) + \mathbf{o}_p \left(e'_p \right) \right) \\
&= \sum_{p \in P} \ell_p \left(\mathbf{o}_p \left(e_p \right) \right) + \ell_p \left(\mathbf{o}_p \left(e'_p \right) \right) \\
&= \sum_{p \in P} \ell_p \left(\mathbf{o}_p \left(e_p \right) \right) + \sum_{p \in P} \ell_p \left(\mathbf{o}_p \left(e'_p \right) \right) \\
&= \ell \left(\mathbf{o} \left(e \right) \right) + \ell \left(\mathbf{o} \left(e' \right) \right)
\end{aligned}
$$

3.3 ALGEBRAIC EQUATIONS AND INEQUALITIES

In this section, we recall *algebraic equations*, *inequalities*, and their *solutions* [Rei13]. We show that solution sets carry over when considering *specializations* of the congruence, but validity does not.

As visualized in Figure 18 (Page 63), each algebraic equation and inequality consists of an abstraction query and a constant.

Definition 29 (Algebraic Equation and Inequality, Solution)
Let $\mathbf{k}$ be a distributed abstraction query. Let $r \in \mathbb{Z}\langle \frac{\emptyset}{F} \rangle$ be a constant.

Then, $\mathbf{k}(P) \doteq r$ is an *algebraic equation*, and $\mathbf{k}(P) \dot{\geq} r$ is an *algebraic inequality*.

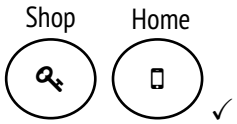
For a parameterized effect $e \in \mathbb{Z}\langle \frac{\mathbb{V}}{F} \rangle^P$, let $\mathbf{k}(e) \equiv r$. Then, e *satisfies* $\mathbf{k}(P) \doteq r$, and e is a *solution* of $\mathbf{k}(P) \doteq r$. Accordingly

F_{23}: 🔑 : B, 📱 : B

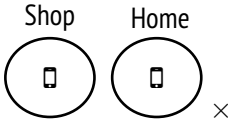
(a) Data scheme $(\mathsf{F}_{23}, \equiv_\emptyset)$

$-1 \cdot \mathsf{Shop} - 1 \cdot \mathsf{Home} \geqslant -[🔑,📱]$

(b) Algebraic inequality I_{23}.



(c) Key and mobile phone at different locations.



(d) The mobile phone at two locations.

Figure 23.: Users in shops or at home: Mutual exclusion as an example for data integrity.

> let $\mathbf{k}(e) \geqq \mathrm{r}$. Then, e *satisfies* $\mathbf{k}(\mathsf{P}) \dot{\geqq} \mathrm{r}$, and e is a *solution* of $\mathbf{k}(\mathsf{P}) \dot{\geqq} \mathrm{r}$.
>
> Let Φ be an algebraic equation or inequality. We denote the *set of all solutions of* Φ by $\mathbb{L}(\Phi)$.

An example of an algebraic inequality is shown in Figure 22d. We studied the data scheme $(\mathsf{F}_{22}, \equiv_{\mathsf{A}_{22}})$ and the query $\mathbf{k}$ in Section 3.2.3. The symbol P_{22} in the inequality is merely syntactical; it can be seen as the unknown of the equation as we use the symbol for the set of places. We observe that $\mathsf{m}^{22.1}$ is a solution: As discussed before, it holds that $\mathbf{k} \odot \mathsf{m}^1 \equiv_\mathsf{A} [🚲_{\mathsf{s}(0)}] \geqq_\mathsf{A} [\,]$. Accordingly, $\mathsf{m}^{22.1}$ is a solution of $\mathbf{k} \odot \mathsf{P}_{22} \dot{\geqq} [\,]$.

In contrast to that, $\mathsf{m}^{22.2}$ is not a solution. The result of applying $\mathbf{k}$ is $\mathbf{k} \odot \mathsf{m}^{22.2} \equiv_\mathsf{A} [🚲_{\mathsf{s}(0)}] - [🚲_{\mathsf{s}(\mathsf{s}(0))}] \not\geqq_\mathsf{A} [\,]$.

Thus, the inequality in Figure 22d has the intended semantic: It is satisfied if and only if for every order an according product is in the storage.

In Figure 23, we show an example to specify data integrity by means of mutual exclusion in a distributed system using an algebraic inequality. The data scheme $(\mathsf{F}_{23}, \equiv_\emptyset)$ models two entities: A key 🔑, and a mobile phone 📱. Furthermore, the example uses the locations Shop and Home with $\textsc{Sort}(\mathsf{Shop}) = \textsc{Sort}(\mathsf{Home}) = B$. Data integrity in this setting is the following constraint: Both the key and the mobile phone are either at Home or at the Shop. Whereas the marking in Figure 23c satisfies this condition as the key is in the shop and the mobile phone is at home, the constraint is not satisfied in Figure 23d: The mobile phone 📱 is at two mutually exclusive locations. The algebraic inequality I_{23} specifies this constraint. Both the integer coefficients and the right-hand side are negative in I_{23}. The number of keys or mobile phones on both places is less or equal than 1 if and only if the negation is greater equal than -1.

$$1 \cdot \text{🚲} \geqq [\,]$$

Figure 24.: Ambiguous notation of an algebraic equation over the distributed data scheme $(\{\mathsf{A}, \mathsf{B}\}, \mathsf{F}_{21}, \equiv_\emptyset)$.

By abuse of notation, we extend notions and notations on sets of markings to algebraic equations and inequalities, referring to its set of solutions. For example, we adapt the notion of validity: Each algebraic equation or inequality Φ is valid, if $\mathbb{L}(\Phi)$ is valid.

ALTERNATIVE NOTATION. To increase readability, we introduce an alternative notation for algebraic equations and inequalities. For each place p we denote the integer coefficient ℓ_p and term induced function $\mathbf{o}_p$ as monomial $\ell_p \cdot \mathbf{o}_p$. Then, we write the sum omitting all places with integer coefficient zero. An example is shown in Figure 22e. The abstraction query of the algebraic inequality I_{22} is depicted as the sum of two monomials: $1 \cdot \textbf{Storage}$ and $-1 \cdot \mathsf{prod}(\textbf{Shop})$.

Although the alternative notation may be easier to read, it can be ambiguous. An example, where the notation is ambiguous is shown in Figure 24. The distributed data scheme $(\{\mathsf{A}, \mathsf{B}\}, \mathsf{F}_{21}, \equiv_\emptyset)$ contains two places. In I_{24}, only one integer coefficient is unequal zero, The only shown term-induced function is constant, as it is induced by a ground term. It is ambiguous which integer coefficient is zero. The coefficient 1 is either for A or B.

Hence, in the rest of this thesis, we use the alternative notion whenever it is unambiguous and otherwise rely on the notation using tables.

3.3.1 Specializations of a Congruence

As usual in algebraic specification [ST12] , one may consider *specializations* of each congruence of a data scheme. A specialization is a congruence, which implies the original congruence as formalized in Section A.4.

Solutions of algebraic equation and inequalities carry over when considering specializations, as shown in the following lemma.

Lemma 30 (Solutions are Solutions of specializations of the Congruence)
Let $N_1 = (P, F, \equiv_1)$ and $N_2 = (P, F, \equiv_2)$ be two distributed data schemes such that $\theta \equiv_1 \theta'$ implies $\theta \equiv_2 \theta'$ for all $\theta, \theta' \in \langle \frac{\emptyset}{F} \rangle$. Let Φ_1 be an algebraic equation or inequality over $(P, F, \equiv_1)$ and ϕ_2 the algebraic equation or inequality over $(P, F, \equiv_2)$ with the same abstraction query and right-hand side.

Then, $\mathbb{L}(\Phi_2) \subseteq \mathbb{L}(\Phi_1)$.

Proof of Lemma 30. Let $r, r' \in \mathbb{Z}\langle \frac{\mathbb{V}}{F} \rangle$ with $r \equiv_1 r'$. Let $\theta \in \langle \frac{\emptyset}{F} \rangle$. Then, we have:

$$\begin{aligned}
\sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv_2 \theta'}} r(\theta') &= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv_2 \theta'}} \sum_{\substack{\theta'' \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv_1 \theta'}} r(\theta'') \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv_2 \theta'}} \sum_{\substack{\theta'' \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv_1 \theta'}} r'(\theta'') \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv_2 \theta'}} r'(\theta')
\end{aligned}$$

And thus, we deduce for all polynomials $r, r' \in \mathbb{Z}\langle \frac{\mathbb{V}}{F} \rangle$:

$$r \equiv_1 r' \text{ implies } r \equiv_2 r' \qquad (*)$$

Let $\mathbf{k}$ be an abstraction query, $m \in \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$, $r \in \mathbb{Z}\langle \frac{\emptyset}{F} \rangle$ with $\mathbf{k} \odot m \equiv_1 r$. Then, by $*$, we have that:

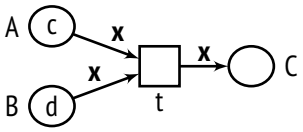
$$\mathbf{k} \odot m \equiv_2 r$$

Thus, if m is a solution of Φ_1, m is also a solution of Φ_2.

However, considering a specialization of a congruence, a transition may induce more steps. An example is shown in Figure 25. Two distributed data schemes $(P_{25.1}, F_{25.1}, \equiv_{A_{25.1}})$ and $(P_{25.2}, F_{25.2}, \equiv_{A_{25.2}})$ are shown in Figure 25a, where $(P_{25.2}, F_{25.2}, \equiv_{A_{25.2}})$ is a specialization of $(P_{25.1}, F_{25.1}, \equiv_{A_{25.1}})$. As $F_{25.1} = F_{25.2}$, both data schemes induce the same two terms: c and d. Moreover, we have that $c \not\equiv_{A_{25.1}} d$, but $c \equiv_{A_{25.2}} d$. Considering the transition t and marking m^{25} shown in Figure 25b, we observe that with respect to $\equiv_{A_{25.1}}$, t does not induce a step from m^{25}. However, with respect to $\equiv_{A_{25.2}}$, t induces a step from m^{25}. In the target marking of that step, the algebraic equation E_{25} is

places P_{25}	function symbols F_{25}	axioms $A_{25.1}$	axioms $A_{25.2}$
A,B,C	$c: \to S$	-	$c \doteq d$
	$d: \to S$		

(a) The distributed data schemes $(P_{25}, F_{25}, \equiv_{A_{25.1}})$ and $(P_{25}, F_{25}, \equiv_{A_{25.2}})$.



(b) Transition t and the marking m^{25}.

$$1 \cdot \mathbf{C} \doteqdot 0$$

(c) Algebraic equation E_{25}.

Figure 25.: Example for a specialization of congruence with a different set of induced steps.

not satisfied. Accordingly, although $\equiv_{A_{25.2}}$ is specialization of $\equiv_{A_{25.1}}$, validity of E_{25} is different with respect to the same transition and the same initial marking.

3.4 DISCUSSION

In this chapter, we introduced algebraic equations and inequalities to specify data integrity in distributed systems modeled by algebraic Petri nets. We discuss restrictions and limitations of our approach in Section 3.4.1 and study related work in Section 3.4.3.

3.4.1 Limitations of Algebraic Equations and Inequalities

Data integrity is an often used and overloaded term in computer science. It is intertwined with the notion of consistency, which is also often used and overloaded. In [DMM15] the terminology around consistency is discussed thoroughly including data integrity. In the sense of [DMM15], data integrity is a synonym for *semantic consistency*, which is defined as follows:

> Semantic consistency is a property of database states. Thus, for a given database schema, semantic consistency can be identified with a subset of all possible states. Ideally, the predicate that corresponds to the characteristic function of that subset is described by an integrity theory [DMM15].

In this section, we list some aspects of data integrity that are not covered by our approach. As we consider formal modeling, we are restricted to model-specific properties, excluding the "real world" such as data quality and physical integrity. Moreover, certain notions of consistency are not covered.

MODELING GAP. In [KK02], the authors separate the *modeling technique* from the *modeling language*: Here, the modeling language is part of the modeling technique; the modeling technique describes *how* to use the modeling language. The modeling technique includes the human factor and interaction with the modeling language. This thesis contributes to the modeling language algebraic Petri nets. However, regarding the gap between the "real world" and the model, the modeling technique is very important. Moreover, it is not possible to cover problems of the "real world" in the modeling language: When inaccurate or invalid data is translated into the modeling language, its invalidity cannot be discovered. On the abstracted level of a model, it is not possible to analyze its relation to the "real world". Hence, all definitions relying on a "real world" such as *data quality* [Orr98], *data accuracy* and *data validity* cannot be tracked with our approach.

However, data integrity in the model can be seen as a necessary, but not sufficient criterion assuming data integrity in the "real world".

PHYSICAL INTEGRITY. Another aspect of data integrity is *physical integrity*, with respect to the underlying file system [SWZ05]. With this perspective, data is not only a logic entity, but also a physical entity, such as in the memory of a computer. Here, physical phenomenas such as radiation can damage the physical data and hence influence physical data integrity. In this thesis, we abstract from such physical integrity and assume physically correct implementations of the models.

CONSISTENCY. As stated, data integrity is intertwined with data consistency. In computer science, the term "consistency" has varying, unclear semantics [DMM15]. Most prominent are their usage in the acronyms *CAP* and *ACID*. We will discuss both separately:

ACID was introduced as an acronym by [HR83] for *atomicity*, *consistency*, *isolation*, and *durability*. Here, consistency means that integrity is preserved by transactions. This way, our definition of data integrity fits. However, in the model, a globally accessible database is assumed, distributed data is not considered. Our approach generalizes this perspective for the distributed setting.

CAP The acronym stands for *consistency*, *availability*, and *persistence* [GL02]. The CAP-theorem states that not all three properties consistency, availability and persistence can be a guaranteed by a distributed system [GL02]. Here, consistency in a distributed system means that the copies of replicated system state are identical to each other at synchronization points. The problem led to different notions of consistency, such as *distributed consistency*, *eventual consistency*, and *partial consistency* [DMM15]. However, all these consistency properties refer to a liveness property or reachability. Our definition of data integrity as a state property does not cover liveness properties or reachability properties.

In the context of the CAP-theorem, every location strives to have a replica of the same global knowledge. It is not considered that different locations may have different (yet related) populations of data entities. In contrast to that, in our approach it is reflected that data may be distributed over a system and not just replicated at different locations.

3.4.2 Relation to Inclusion Dependencies

In classical databases, data integrity is often understood as referential integrity specified by *inclusion dependencies* [AHV95]. An inclusion dependency consists of two queries and is satisfied if the result of the first query is a subset of the result of the second query. As algebraic Petri nets are built on bag semantics in a distributed setting, inequalities can be seen as an adaption of inclusion dependencies. Intuitively, the set inclusion $\subseteq$ corresponds to the partial order $\leqslant$ on bags, where the quantity is taken into account. The difference of two queries may be expressed as an abstraction query. Using the empty bag as the right-hand results in an algebraic inequality. Thus, algebraic inequalities can be seen as "bag-semantics-aware" inclusion dependencies.

However, the properties expressed with inclusion dependencies and algebraic inequalities are different. Using an inclusion dependency over set semantics, it is only required that an according object exists. In the example shown in Figure 22 (Page 67) an inclusion dependency could be interpreted as follows: It is sufficient, if one bicycle exists, no matter how many bicycles are ordered. In contrast to that, algebraic equations and inequalities consider the quantity of objects in a bag: The inequality I_{22} requires sufficiently many bicycles at the storage.

When modeling with Petri nets, it is a basic assumption that data entities may be not unique in the modeled system. As a result modeling relies on bag semantics and algebraic equations and inequalities

take bag semantics into account. In this thesis, we do not consider inclusion dependencies and restrict ourselves to algebraic equations and inequalities for the specification of data integrity.

3.4.3 Related Work

In this section, we discuss related work in two steps: First, we discuss related work regarding the specification of data integrity in distributed systems. Secondly, we discuss related work regarding equations and inequalities in algebraic Petri nets. Both lines of research have been pursued in parallel.

SPECIFYING DATA INTEGRITY IN DISTRIBUTED SYSTEMS. The authors of [CCL13; Cal+04; CCL13] examine distributed data with the goal of data integration. The authors recognize that global data integrity of a distributed data is difficult to handle. In contrast to our work, the goal of data integration striving for global knowledge, whereas we consider a distributed setting.

In literature, data integrity in distributed systems is considered mainly with respect to distributed databases using a relational model [AII09; AIU09; Ord+09; Ibr06; IAU07]. Summarizing, the importance of data integrity is stated, especially in distributed systems. Moreover, guaranteeing integrity constraints is challenging as global knowledge is needed during run time. In [Ord+09], a tool is presented to analyze data integrity over distributed databases using a relational data model. In contrast to the mentioned approaches with queries over a relational model, our model is less expressive, as the class of abstraction queries is restricted. However, the shown examples of referential integrity and mutual exclusion with the introduced class of abstraction queries.

EQUATIONS AND INEQUALITIES IN ALGEBRAIC PETRI NETS. Equations have been a tool of all variants in Petri nets since many years and their usage to prove properties of a distributed system [Vau85; Vau87; MV86; GL79; GLT79; Rei13; Rei91; Rei98; Mur89; Jen81b; Jen81a; TS16b]. Regarding inequality, the field of *traps* and *siphons* are an established tool for specification and analysis tool [Hac72; HGD08; Mur89; Rei13; TS15; TS16a]. Regarding high-level Petri nets fewer methods for specification and analysis based on inequalities have emerged [Sch97; KV98]. Their use to specify referential integrity and mutual exclusion [Rei13] is also well-known.

4 ANALYSIS

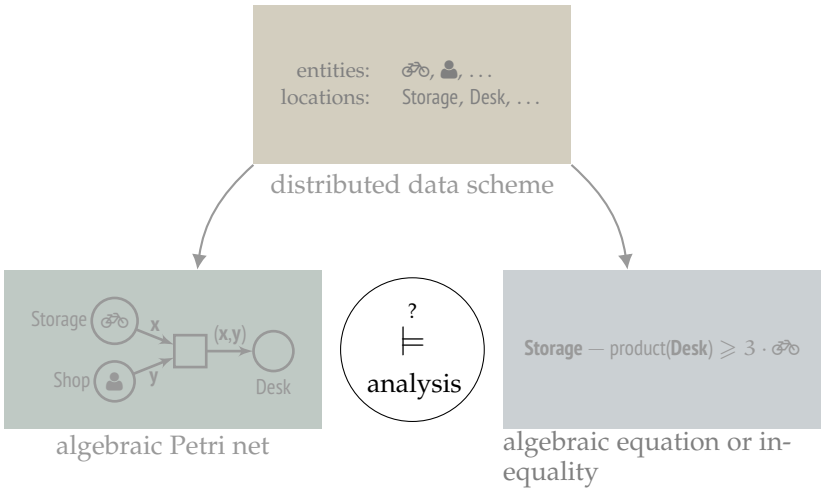


Figure 26.: Overview of part II. The contents of chapter 2 and 3 are grayed out.

We have introduced our formal model for distributed data-aware systems and data integrity in the Chapters 2 and 3. In this chapter we explore analysis and verification opportunities of data integrity specified by an algebraic equation or inequality in an algebraic Petri net. In Section 4.1, we show the undecidability of validity of an algebraic equation and inequality in algebraic Petri nets. In Section 4.2, we recall the class of *stable* equations and inequalities in an algebraic Petri net structure. An algebraic equation or inequality I is stable if all steps of the transitions *preserve* satisfaction of I. In Section 4.2.1, we show how *non-preserving steps* can be used as a counterexample for stability or validity of an algebraic equation or inequality.

4.1 GENERAL UNDECIDABILITY

In this section, we show that validity of an algebraic equation and inequality in an algebraic Petri net over a compact data scheme is undecidable. Our approach is sketched in Figure 27. We show undecidability in two steps: First, reducing the undecidable problem of

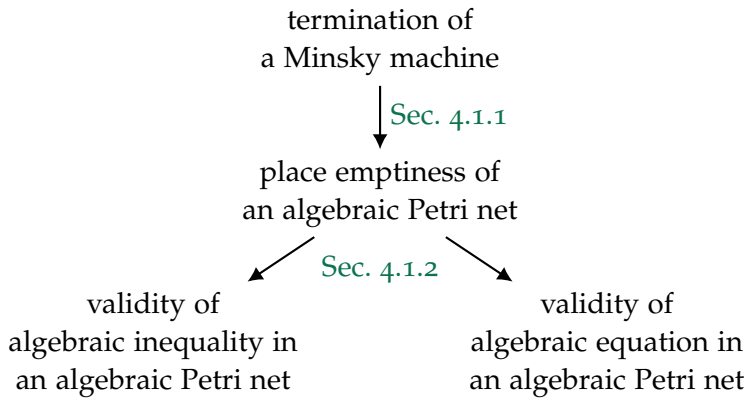


Figure 27.: The reductions shown in this section.

termination of a *Minsky machine* to the problem of *place emptiness* of an algebraic Petri net. Second, we show that place emptiness can be reduced to validity of an algebraic equation or inequality.

Termination of a Minsky Machine

In this section, we recall the definition of a Minsky machine and the problem of termination, which has been shown to be undecidable. In the following section we will reduce this problem to place emptiness of an algebraic Petri net.

A Minsky machine consists of a number of registers and a sequence of instructions. Each instruction is either an increment instruction, a decrement/jump-zero instruction, or a halt instruction. The last instruction is always a halt instruction.

Definition 31 (Minsky Machine)
Let $R \in \mathbb{N} \setminus \{0\}$ be the *number of registers*. Let $h \in \mathbb{N} \setminus \{0\}$ be the *number of instructions*. Let $\ell, \ell', r \in \mathbb{N}$ with $\ell, \ell' \leqslant h$ and $r \leqslant R$.

Then, we define the following instructions:

- $\texttt{Inc}(r, \ell)$ is an *increment instruction*.
- $\texttt{DecJumpZero}(r, \ell, \ell')$ is a *decrement/jump-zero instruction*.
- $\texttt{Halt}$ is the *halt instruction*.

Let $I = (I_1, \ldots, I_{h-1}, \texttt{Halt})$ be a *sequence of instructions*.

Then, (R, I) is a *Minsky machine* with h instructions.

In Figure 28, three examples of Minsky machines are shown. Each of them has one register and four instructions.

A state of a Minsky machine assign a natural number to each register and contains the current instruction number.

Steps between states are defined based on the instructions. Intuitively, a `Halt` instruction halts the Minsky machine and no further step is possible. An $\texttt{Inc}(r, \ell)$ instruction increments the value stored in register r and jumps to instruction ℓ. A $\texttt{DecJumpZero}(r, \ell, \ell')$ instruction has the following semantics: If the value of register r is greater or equal to one, the value of r is decremented and the next instruction is ℓ. Otherwise, the next instruction is ℓ'.

Definition 32 (States and Steps of a Minsky Machine)
Let (R, I) be a Minsky machine with h instructions. Let $(\rho, \ell) \in \mathbb{N}^R \times \{1, \ldots, h\}$.

Then, (ρ, ℓ) is a *state* of (R, I).

Let (ρ, ℓ) and (ρ', ℓ') be states of (R, I), and one of the following conditions be true:

1. $I_\ell = \texttt{Inc}(r, \ell')$ and for $1 \leqslant i \leqslant R$, it holds that
$$\rho'_i = \begin{cases} \rho_i + 1 & \text{, if } r = i \\ \rho_i & \text{, otherwise.} \end{cases}$$
2. $I_\ell = \texttt{DecJumpZero}(r, \ell', \ell'')$, and $\rho_r \geqslant 1$, and for $1 \leqslant i \leqslant R$, it holds that
$$\rho'_i = \begin{cases} \rho_i - 1 & \text{, if } r = i \\ \rho_i & \text{, otherwise.} \end{cases}$$
3. $I_\ell = \texttt{DecJumpZero}(r, \ell'', \ell')$, and $\rho_r = 0$, and $\rho' = \rho$.

Then, $(\rho, \ell) \to (\rho', \ell')$ is a *step of* (R, I). The reflexive transitive closure of the set of steps $\to^*$ is the *reachability* relation.

Based on the steps and reachability, we define the problem of termination of a Minsky machine. Intuitively, at the initial state, every register has the value zero, and the machine starts with the first instruction. From thereon other states are reachable according to the steps. If the last halt instruction is reachable with arbitrary values for the registers, the Minsky machine terminates.

1: Inc(1,2)	1: Inc(1,2)	1: Inc(1,2)
2: Inc(1,3)	2: DecJumpZero(1,3,2)	2: DecJumpZero(1,3,4)
3: Inc(1,4)	3: DecJumpZero(1,3,4)	3: DecJumpZero(1,4,1)
4: Halt	4: Halt	4: Halt
(a) terminating	**(b)** terminating	**(c)** not terminating

Figure 28.: Three Minsky machines.

Definition 33 (Termination of a Minsky Machine)
Let (R, I) be a Minsky machine with h instructions. Let $\rho \in \mathbb{N}^R$ such that $(0, \ldots, 0, 1) \to^* (\rho, h)$.
Then, (R, I) is *terminating*.

In Figure 28, the first two of the shown Minsky machines are terminating, whereas the third is not. The first Minsky machine, as shown in Figure 28a, increases register 1 three times and then terminates. The second Minsky machine, as shown in Figure 28b, increases register 1, then decreases it back and then jumps to the halt instruction and terminates. The third Minsky machine, as shown in Figure 28c, does not terminate: It increases the value in register 1, then decreases and jumps back to instruction 1 and starts again. The last instruction of the third Minsky machine is not reachable. Hence, the third Minsky machine is not terminating.

It is known that the problem of termination is undecidable:

Lemma 34 (Termination of a Minsky Machine is Undecidable [Min67])
Let M be a Minsky machine.
Then, termination of M is undecidable.

Place Emptiness in an Algebraic Petri Net

In this paragraph, we define the place emptiness problem of an algebraic Petri net: A place of an algebraic Petri net is empty, if the place is assigned the empty bag by all reachable markings.

Definition 35 (Place Emptiness)
Let $\left(P, F, \equiv, T, m^0\right)$ be an algebraic Petri net. Let $p \in P$. Let for all $m \in \textsc{Reach}_T(m^0)$ hold: $m_p = [\,]$.
Then, $\left(P, F, \equiv, T, m^0\right)$ is *place empty in* p, or p-*empty*.

function symbols (FN)	axioms	terms
0: $\to N$	-	0, s(0), s(s(0)), ...
s: $N \to N$		

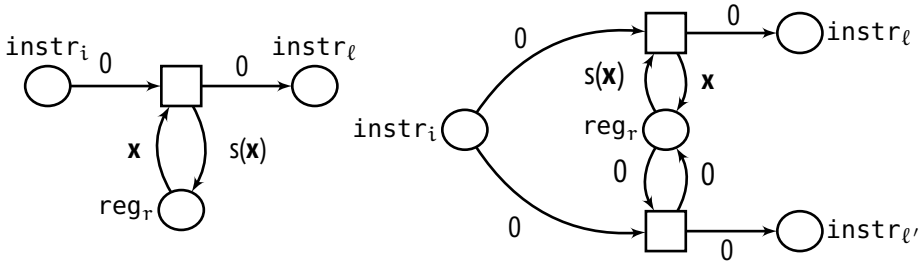
Figure 29.: The data scheme $(\mathsf{FN}, \equiv_\emptyset)$ modeling natural numbers.

A very basic example of an algebraic Petri net that is p-empty is given by the following two conditions: First, for each transition t we have $t_p^+ = [\,]$ and secondly for the initial marking m^0 we have $m_p^0 = [\,]$. Then, p is empty in the initial marking and no transition may produce a term on p and hence it stays empty in all reachable markings.

4.1.1 Reduction: The Termination of a Minsky Machine to Place Emptiness

In this section, we reduce the problem of termination of a Minsky machine to the problem of place emptiness of an algebraic Petri net. To this end, we encode a given Minsky machine into an algebraic Petri net over a compact data scheme. We show an equivalence between the steps of the Minsky machine and its encoding. Based thereon we reduce termination to place emptiness.

For the encoding, we fix the compact data scheme $(\mathsf{FN}, \equiv_\emptyset)$ that consists of two function symbols and the identity as congruence as shown in Figure 29. We used this data scheme before in Chapters 2 and 3 to model natural numbers. Intuitively, $(\mathsf{FN}, \equiv_\emptyset)$ is a simple data scheme with an infinite number of terms. We encode a given Minsky machine into an algebraic Petri net over the data scheme $(\mathsf{FN}, \equiv_\emptyset)$ in two steps: First, we encode the structure into a distributed data scheme and states into markings. Second, we encode instructions into transitions. The idea is the following: Every register r is encoded as a place $\mathtt{reg}_r$. We map the value x of each register to the term θ_x that models the natural number x: zero as 0, one as s(0), etc. Moreover, every instruction I_i is encoded as a place $\mathtt{instr}_i$. The idea is, that a marking assigns $[0]$ to the place $\mathtt{instr}_i$, if and only if the encoded state of the Minsky machine is currently at the instruction I_i. Here, the term 0 is used as a token. For the instruction places, the term 0 does not model a natural number.



(a) Encoding of an instruction $I_i = \texttt{Inc}(r, \ell)$.

(b) Encoding of an instruction $I_i = \texttt{DecJumpZero}(r, \ell, \ell')$.

Figure 30.: Encoding instructions of a Minsky machine into transitions of an algebraic Petri net according to Definition 37.

Definition 36 (Encoding of a State of a Minsky Machine)
Let $M = (R, I)$ be a Minsky machine with h instructions and

$$P = \{\texttt{reg}_r \mid 1 \leqslant r \leqslant R\} \cup \{\texttt{instr}_i \mid 1 \leqslant i \leqslant h\}.$$

Then, $(P, FN, \equiv_\emptyset)$ is the *encoded distributed data scheme of* M.

Let $(\rho, \ell) \in \mathbb{N}^R \times \{1, \ldots, h\}$ be a state of M. For $x \in \mathbb{N}$, we define $\theta_x \in \langle {}^{\emptyset}_{FN} \rangle$ by

$$\theta_x = \begin{cases} 0 & \text{, if } x = 0 \\ s(\theta_{x-1}) & \text{, otherwise.} \end{cases}$$

Then, we define the marking $m^{(\rho,\ell)} \in \mathbb{N}\langle {}^{\emptyset}_{FN} \rangle^P$ of $(P, FN, \equiv_\emptyset)$ as follows for $p \in P$ and $\theta \in \langle {}^{\emptyset}_{FN} \rangle$:

$$m_p^{(\rho,\ell)} = \begin{cases} [0] & \text{, if } p = \texttt{instr}_\ell \\ [\theta_{\rho(r)}] & \text{, if } p = \texttt{reg}_r \\ [\,] & \text{, otherwise.} \end{cases}$$

The encoding of states into markings is not surjective. Hence, there exists markings of the encoded algebraic Petri net, which do not have an according state of the Minsky machine.

Now, we encode the instructions of a Minsky machine into transitions. For every increment instruction $\texttt{Inc}(r, \ell)$ a transition is introduced as shown in Figure 30a: The term of the corresponding register r is increased using the s function symbol. The term 0 is "moved" from the current instruction place $\texttt{instr}_i$ to the next instruction place

$\texttt{instr}_\ell$. For every $\texttt{DecJumpZero}(r, \ell, \ell')$ two transitions are introduced as shown in Figure 30b: One for the "decrease" case, which is symmetrical to increase construction for a term unequal 0 on the place $\texttt{reg}_i$. The other transition is only enabled, if a marking assigns the term 0 to the place of the corresponding register $\texttt{reg}_r$. Furthermore, we consider the initial marking that encodes the state $(0, \ldots, 0, 1)$.

Definition 37 (Encoding of Minsky Machine)
Let $M = (R, I)$ be a Minsky machine with h instructions, Let $N(M) = \left(P, FN, \equiv_\emptyset, T, m^{(0,\ldots,0,1)}\right)$ be the algebraic Petri net such that $\left(P, FN, \equiv_\emptyset\right)$ is the encoded distributed data scheme of M, $m^{(0,\ldots,0,1)}$ encodes the state $(0, \ldots, 0, 1)$. Moreover, we have $T = T_1 \cup \cdots \cup T_h$ such that for all $1 \leqslant i \leqslant h$ holds:

- $I_i = \texttt{Inc}(r, \ell)$ if and only if $T_i = \{t\}$ such that:

$$t_p^- = \begin{cases} [0] & \text{, if } p = \texttt{instr}_i \\ [\mathbf{x}] & \text{, if } p = \texttt{reg}_r \\ [\,] & \text{, otherwise.} \end{cases}$$

$$t_p^+ = \begin{cases} [0] & \text{, if } p = \texttt{instr}_\ell \\ [s(\mathbf{x})] & \text{, if } p = \texttt{reg}_r \\ [\,] & \text{, otherwise.} \end{cases}$$

- $I_i = \texttt{Inc}(r, \ell)$ if and only if $T_i = \{s, t\}$ such that:

$$t_p^- = \begin{cases} [0] & \text{, if } p = \texttt{instr}_i \\ [s(\mathbf{x})] & \text{, if } p = \texttt{reg}_r \\ [\,] & \text{, otherwise.} \end{cases}$$

$$t_p^+ = \begin{cases} [0] & \text{, if } p = \texttt{instr}_\ell \\ [\mathbf{x}] & \text{, if } p = \texttt{reg}_r \\ [\,] & \text{, otherwise.} \end{cases}$$

$$s_p^- = \begin{cases} [0] & \text{, if } p = \texttt{instr}_i \\ [0] & \text{, if } p = \texttt{reg}_r \\ [\,] & \text{, otherwise.} \end{cases}$$

$$s_p^+ = \begin{cases} [0] & \text{, if } p = \texttt{instr}_{\ell'} \\ [0] & \text{, if } p = \texttt{reg}_r \\ [\,] & \text{, otherwise.} \end{cases}$$

- $I_i = \texttt{Halt}$ if and only if $T_i = \emptyset$.

Then, $N(M)$ encodes M.

As a first observation, we show in Lemma 38 that the encoding of a given Minsky machine into an algebraic Petri net is computable.

Lemma 38 (Encoding is Computable)
Let M be a Minsky machine. Then, the encoding in an algebraic Petri net is computable.

Proof of Lemma 38. We first observe that $(\mathsf{FN}, \equiv_\emptyset)$ is finitely representable. Then, as the number of instructions is finite, the number of places is finite. The number of transitions is also finite, as for each instruction at most two transitions are generated. Each entry of the transitions is again a finite polynomial. Hence, the encoding of a Minsky machine is computable.

In the rest of this section, we will show that a Minsky machine does not terminate if and only if its encoding is $\texttt{instr}_h$-empty. That means that no reachable marking of the encoding represents the halt instruction.

Now, we can relate the steps of a Minsky Machine M to the steps of the encoding $N(M)$. We published a shorter version of the proof in [TS16b; TS16c]

Lemma 39 (Equivalence of Steps)
Let $M = (R, I)$ be a Minsky machine with h instructions and $(\rho, \ell), (\rho', \ell') \in \mathbb{N}^R \times \{0, \dots, h\}$ be states of M. Let $N(M) = \left(P, \mathsf{FN}, \equiv_\emptyset, T, m^0\right)$ be the encoded algebraic Petri net of M and $m, m' \in \mathbb{N}\langle {}^{\emptyset}_{\mathsf{FN}} \rangle^P$ the encoded states of (ρ, ℓ) and (ρ', ℓ'), respectively.

Then, the following two statements are equivalent:

1. $(\rho, \ell) \to (\rho', \ell')$ is a step of M.
2. $(m, m') \in \mathcal{R}(T)^\uparrow$.

Proof of Lemma 39. 1.⇒2.: As there exists a step, I_ℓ is either an increment or a decrement/jump-zero statement. For the latter, the instruction either decreases the value of a register or jumps to another instruction, if the register has value zero in that state. In every case, ℓ' is the target instruction of each instruction. We distinct the three cases.

1st case: $I_\ell = \texttt{Inc}(r, \ell')$.
By Definition 37, there exists $t \in T$ with $t^-_{\texttt{instr}_\ell} = t^+_{\texttt{instr}_{\ell'}} = [0]$, $t^-_{\texttt{reg}_r} = [\mathbf{x}]$ and $t^+_{\texttt{reg}_r} = [s(\mathbf{x})]$. As m encodes (ρ, ℓ), we have that $m_{\texttt{reg}_r} = [\theta]$ for some $\theta \in \langle {\emptyset \atop \mathsf{FN}} \rangle$ representing a natural number. Accordingly, $m \in \mathcal{R}\left(t^-\right)^\uparrow$. Thus, let $m = \underline{m} + \sigma\left(t^-\right)$ for some $\underline{m} \in \mathbb{N}\langle {\emptyset \atop \mathsf{FN}} \rangle^P$ and $\sigma \in \{\mathbf{x}\} \overset{F}{\rightsquigarrow} \emptyset$. It remains to show that $m' = \underline{m} + \sigma\left(t^+\right)$. To this end, we observe that ρ and ρ' are only different in r and hence m and m' are only different in $\texttt{reg}_r$, $\texttt{instr}_\ell$, nd $\texttt{instr}_{\ell'}$. We have $m'_{\texttt{reg}_r} = [s(\theta)] = \sigma\left(t^+_{\texttt{reg}_r}\right)$ and $\sigma\left(t^+_{\texttt{instr}_{\ell'}}\right) = [0]$ and we deduce that $m' = \underline{m} + \sigma\left(t^+\right)$. Hence, $(m, m') \in \mathcal{R}(t)^\uparrow$ is a step.

2nd case: $I_\ell = \texttt{DecJumpZero}(r, \ell', \ell'')$ and $\rho_r > 0$.
The reasoning is the same as in the first case, except that $t^-_{\texttt{reg}_r} = [s(\mathbf{x})]$, and $t^+_{\texttt{reg}_r} = [\mathbf{x}]$. As $\rho_r > 0$, we have that the term on $m_{\texttt{reg}_r}$ is unifiable with $s(\mathbf{x})$. Again, this implies $(m, m') \in \mathcal{R}(t)^\uparrow$.

3rd case: $I_\ell = \texttt{DecJumpZero}(r, \ell'', \ell')$ and $\rho_r = 0$.
In this case, we have $m_{\texttt{instr}_\ell} = [0]$ and $m_{\texttt{reg}_r} = [0]$. By Definition 37 there exists a transition $t \in T$ with $t^-_{\texttt{reg}_r} = t^+_{\texttt{reg}_r} = [0]$ and $t^-_{\texttt{instr}_\ell} = t^+_{\texttt{instr}_{\ell''}} = [0]$. And thus, $m \in \mathcal{R}\left(t^-\right)^\uparrow$. We can ignore the firing mode, as $\mathbb{V}(t) = \emptyset$ and thus $\mathcal{R}(t) = \{t\}$. It remains to show that $m' = m - t^- + t^+$. As we have $\rho = \rho'$, we deduce $m'_{\texttt{reg}_i} = 0$ for all $1 \leqslant i \leqslant R$. As $t^-_{\texttt{reg}_i} - t^+_{\texttt{reg}_i} = 0$ for all $1 \leqslant i \leqslant R$, we deduce $m'_{\texttt{reg}_i} = m_{\texttt{reg}_i}$. And moreover, $m'_{\texttt{instr}_{\ell'}} = [0] = t^+_{\texttt{instr}_{\ell'}}$ and we deduce $(m, m') \in \mathcal{R}(t)^\uparrow$.

2.⇒1.: As (m, m') is a step, there exists a transition t with $m \in \mathcal{R}\left(t^-\right)^\uparrow$. By Definition 37, it follows that: $t^-_{\texttt{instr}_\ell} \geqslant [0]$. According to Definition 37, then either term on a place encoding a register is increased, decreased or no place encoding a register is affected. We distinguish the three cases:

1st case: There exists $r \leqslant R$ with $m'_{\mathsf{reg}_r} = s(m_{\mathsf{reg}_r}) = [s(\theta)]$ (for a $\theta \in \langle {}^{\emptyset}_{\mathsf{FN}} \rangle$).
Then, $I_\ell = \mathtt{Inc}(r, \ell')$. We observe:

$$m_i = \begin{cases} [s(\theta_{\rho_i})] & \text{, if } i = r \\ [\theta_{\rho_i}] & \text{, otherwise.} \end{cases}$$

Moreover, $m'_{\ell'} = [0]$ as $t^+_{\ell'} = [0]$. Accordingly, m' encodes (ρ', ℓ') with:

$$\rho'_i = \begin{cases} \rho_i + 1 & \text{, if } i = r \\ \rho_i & \text{, otherwise.} \end{cases}$$

Thus, (ρ', ℓ') is the result of applying $\mathtt{Inc}(r, \ell')$ to (ρ, ℓ) and hence, $(\rho, \ell) \to (\rho', \ell')$ is a step.

2nd case: There exists $r \leqslant R$ with $s(m'_{\mathsf{reg}_r}) = m_{\mathsf{reg}_r} = [s(\theta)]$ (for a $\theta \in \langle {}^{\emptyset}_{\mathsf{FN}} \rangle$).
Then, we have $\rho_r > 0$ and $I_\ell = \mathtt{DecJumpZero}(r, \ell', \ell'')$ for some $1 \leqslant \ell'' \leqslant h$. Symmetrically to the first place, we observe that m' encodes (ρ', ℓ') with

$$\rho'_i = \begin{cases} \rho_i - 1 & \text{, if } i = r \\ \rho_i & \text{, otherwise.} \end{cases}$$

Thus, (ρ', ℓ') is the result of applying $\mathtt{DecJumpZero}(r, \ell', \ell')$ to (ρ, ℓ) and hence, $(\rho, \ell) \to (\rho', \ell')$ is a step.

3rd case: $m'_{\mathsf{reg}_r} = m_{\mathsf{reg}_r}$ for all $r \leqslant R$.
Then, $I_\ell = \mathtt{DecJumpZero}(r, \ell'', \ell')$ for some $r \leqslant R$ and some $\ell'' \leqslant h$. By Definition 37 follows $(m, m') \in \mathcal{R}(t)^\uparrow$ for a transition with $t^-_{\mathsf{reg}_r} = t^+_{\mathsf{reg}_r} = [0]$. Moreover, $m'_{\ell'} = [0]$ as $t^+_{\ell'} = [0]$. Accordingly, m' encodes (ρ, ℓ'). Thus, (ρ', ℓ') is the result of applying $\mathtt{DecJumpZero}(r, \ell', \ell')$ to (ρ, ℓ) and hence, $(\rho, \ell) \to (\rho', \ell')$ is a step.

Using the above lemma, we prove the following Lemma 40: a Minsky machine terminates if and only if its encoding into an algebraic Petri net is place empty for the place representing the `Halt` instruction.

Lemma 40 (Reduction: Termination to Place Emptiness)
Let M be a Minsky machine and N_M its encoding.
Then, the following statements are equivalent:

1. M does not terminate.
2. N_M is $\mathtt{instr}_h$-empty.

Proof of Lemma 40. 1.⇒2.: Let m be a reachable marking of $N(M)$. Applying Lemma 39 inductively, we deduce that m encodes a state (ρ, ℓ) that is reachable from $(0, \dots, 0, 1)$ in M. As M does not terminate, it follows that $\ell \neq h$ and hence $m_{\mathtt{instr}_h} = 0$. Thus, $N(m)$ is $\mathtt{instr}_h$-empty.

2.⇒1.: Let (ρ, ℓ) be a reachable state of M. Applying Lemma 39 inductively, we deduce that there exists a reachable marking m that encodes (ρ, ℓ). As $N(M)$ is $\mathtt{instr}_h$-empty, it holds that $m_{\mathtt{instr}_h} = 0$. Hence, it holds that $\ell \neq h$. As this holds for all reachable states, M does not terminate.

To conclude this section, we reduce termination of a Minsky machine to place emptiness and deduce that place emptiness in an algebraic Petri net is undecidable.

Lemma 41 (Place Emptiness is Undecidable)
Place emptiness in an algebraic Petri net over the data scheme $(\mathsf{FN}, \equiv_\emptyset)$ is undecidable.

Proof of Lemma 41. We prove Lemma 41 indirect: Assuming place emptiness is decidable by some procedure O, we describe how to decide termination of a Minsky machine:

We encode an arbitrary Minsky machine M into $N(M)$, which is computable by Lemma 38. We decide $\mathtt{instr}_h$-emptiness using O. If O returns false, M terminates. If O return true, M does not terminate.

This procedure is correct by Lemma 40. Hence, we can decide termination of a Minsky machine, which contradicts Lemma 34.

4.1.2 Reduction: Validity to Place Emptiness

In this section, we reduce the problem of place emptiness in an algebraic Petri net to validity of an algebraic equation or inequality. To this end, we define the abstraction query that only considers one

place p:	x	y	z
$\mathbf{o}_p$:	**x**	**y**	**z**
ℓ_p:	0	−1	0

Figure 31.: A negative unary abstraction query $(\mathbf{o}, \ell)$ over y.

place with coefficient −1 and the identity as operation. For a place q, the result of a *negative unary abstraction query over* q applied to a marking is the bag of terms assigned to q by that marking, multiplied with −1.

Definition 42 (Negative Unary Abstraction Query)
Let $\left(P, F, \equiv, T, m^0\right)$ be an algebraic Petri net. Let $q \in P$. Let $\mathbf{o} \in \langle {}^{P}_{F} \rangle^P$ be with $\mathbf{o}_p = p$ for all $p \in P$. Let $\ell \in \mathbb{N}^P$ be with $\ell_q = -1$ and $\ell_p = 0$ for all $p \in P \setminus \{q\}$.
Then, $(\mathbf{o}, \ell)$ is a *negative unary abstraction query over* q.

A negative unary abstraction query over y is shown in Figure 31 using the places x, y, and z.

Using a unary abstraction query, we can construct an algebraic equation or inequality, such that they are satisfied if and only if the according place is empty.

Lemma 43 (Unary Equation is Equivalent to Place Emptiness)
Let $\left(P, FN, \equiv_\emptyset, T, m^0\right)$ be an algebraic Petri net. Let $q \in P$. Let $\mathbf{k} = (\mathbf{o}, \ell)$ be the negative unary abstraction query over q. Let $m \in \mathbb{N}\langle {}^{\emptyset}_{FN} \rangle^P$.
Then, the following statements are equivalent:

1. $\mathbf{k} \odot m \equiv_A [\,]$
2. $\mathbf{k} \odot m \geqq_A [\,]$
3. $m_q = [\,]$

Proof of Lemma 43. First, we observe that the congruence $\equiv_\emptyset$ is equality, hence $\equiv_\emptyset$ and $\geqq_\emptyset$ reduce to $=$ and $\geqslant$, respectively. As all integer coefficients except ℓ_q are equal to zero, we observe:

$$\begin{aligned} \mathbf{k} \odot m &= \sum_{p \in P} \ell_q \mathbf{o}_p(m_p) \\ &= \ell_q \mathbf{o}_q(m_q) \end{aligned}$$

$$= -1\mathbf{o}_q(m_q)$$
$$= -1m_q \qquad (*)$$

Accordingly, $*$ implies the equivalence of 1.) and 3.). The implication of 1). to 2.) follows as $0 \geqslant -0 = 0$. It remains to show:

2.⇒1.: As $m_q \geqslant [\,]$, it follows that $-1m_q \leqslant [\,]$. Hence $[\,] \leqslant m_q \leqslant [\,]$ implies $m_q = [\,]$.

Thus we can reduce the problem of place emptiness to validity of an algebraic equation or inequality and conclude that validity is undecidable.

Theorem 44 (Validity of an Algebraic Equation or Inequality is Undecidable)
Let $N = \left(P, F, \equiv, T, m^0\right)$ be an algebraic Petri net structure and $(F, \equiv)$ be compact. Let Φ be an algebraic equation or inequality.
Then, validity of Φ in N is undecidable.

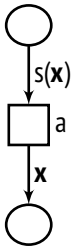
Proof of Theorem 44. We prove this theorem indirect. Let O_E and O_I be procedures to decide validity of an algebraic equation and inequality. Then, we can decide place emptiness for algebraic Petri net over $\left(FN, \equiv_\emptyset\right)$ with the following procedure we sketch:

Given an algebraic Petri net over $\left(FN, \equiv_\emptyset\right)$ and the place q, we compute the negative unary abstraction query over q: $\mathbf{k} = (\mathbf{o}, \ell)$. Then, we use either O_E to decide validity of $\mathbf{k} \odot P \doteq [\,]$, or O_I to decide validity of $\mathbf{k} \odot P \dot{\geqq} [\,]$.

By Lemma 43, the result is valid if and only if q is empty for all reachable markings. Thus, if either validity of an algebraic equation or inequality is decidable, then place emptiness is decidable. This is a contradiction to Lemma 41.

4.2 STABILITY

In the previous section, we showed that validity of an algebraic equation and inequality is undecidable. In this section, we show a sufficient criterion for validity. We recall the concept of *stability* for algebraic Petri nets. Stability is an established concept in computer science [Flo67; Dij80; Hoa72] and not specific to algebraic Petri nets. In the literature, stability is also described by the terms "invariant", "inductive invariant", "loop invariant" and others. However, as the



(a) S

marking	A	B
$m^{32.1}$	0	[0]
$m^{32.2}$	[s(0)]	0
$m^{32.3}$	[0]	0

(b) Markings of S.

set of markings	stable ?
$\{m^{32.1}\}$	✓
$\{m^{32.2}\}$	×
$\{m^{32.1}, m^{32.2}\}$	✓
$\{m^{32.3}\}$	✓

(c) Stable and unstable sets of markings.

Figure 32.: An algebraic Petri net structure S over the data scheme $(\mathsf{FN}, \equiv_\emptyset)$ from Figure 29, with some stable and unstable sets of markings, respectively.

terminology of invariants is overloaded in the literature on Petri nets, we avoid the term "invariant" and use the term stability following the terminology of [Rei13].

Definition 45 (Stability)
Let $S = (P, F, \equiv, T)$ be an algebraic Petri net structure, $\left(S, m^0\right)$ be an algebraic Petri net, and $\phi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$. Let each $(m, m') \in \mathcal{R}(T)^\uparrow$ satisfy:

$$m \in \phi \text{ implies } m' \in \phi\,.$$

Then, ϕ is *stable* in both, S and $\left(S, m^0\right)$.

We observe that stability is defined solely based on steps. Accordingly, stability in algebraic Petri nets can be seen as a special case for the definition of stability. In general, stability may be defined with respect to an arbitrary binary relation. However, as we only discuss stability for algebraic Petri nets, we restrict ourselves to that case.

Figure 32 illustrates stability of an algebraic Petri net structure with an example. The used data scheme $(\mathsf{FN}, \equiv_\emptyset)$ models the natural numbers as shown in Figure 29. The distributed data scheme contains two places A and B, and one transition a. Moreover, markings $m^{32.1}$,$m^{32.2}$,$m^{32.3}$, and four subsets of $\{m^{32.1}, m^{32.2}, m^{32.3}\}$ are shown. We turn to Figure 32c: We observe that no step of the transition a starts from $m^{32.1}$. Hence, the set $\{m^{32.1}\}$ is stable in S. Moreover, every marking without a term on A is stable. We observe that $(m^{32.2}, m^{32.1}) \in \mathcal{R}(a)^\uparrow$ is a step of S. Accordingly, the singleton set $\{m^{32.2}\}$ is not stable in S, as $m^{32.1} \notin \{m^{32.2}\}$. However, the set $\{m^{32.1}, m^{32.2}\}$ is stable. For

$m^{32.3}$, again the transition a does not induce a step starting from $m^{32.3}$ as $s(x)$ and 0 are not unifiable. Accordingly, $\{m^{32.3}\}$ is stable in S.

Stability induces the following important lemma: A stable set of markings ϕ subsumes the reachable markings if and only if the initial marking is in ϕ.

Lemma 46 (Stability Reduces to Validity in the Initial Marking) Let $N = \left(P, F, \equiv, T, m^0\right)$ be an algebraic Petri net and $\phi \subseteq \mathbb{N}\langle {\emptyset \atop F} \rangle^P$ be stable.

Then, the following statements are equivalent:

1. $\textsc{Reach}_T(m^0) \subseteq \phi$
2. $m^0 \in \phi$

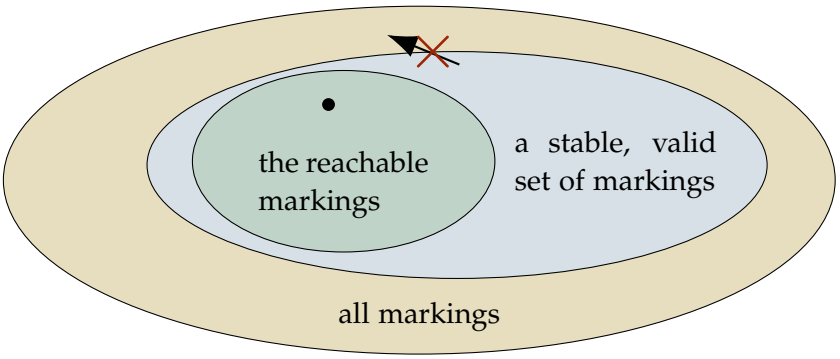
Proof of Lemma 46. 1.⇒2.: As $m^0 \in \textsc{Reach}_T(m^0)$, we have $m^0 \in \phi$.

2.⇒1.: Let $m^0, \ldots, m^n \in \textsc{Reach}_T(m^0)$ with $\left(m^i, m^{i+1}\right) \in \mathcal{R}(T)^\uparrow$ for $0 \leqslant i < n$. By Definition 45, we may apply induction and deduce $m^n \in \phi$.

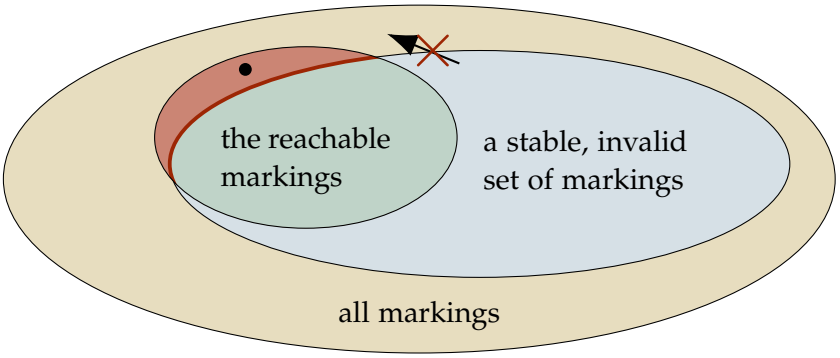
The lemma is visualized as two Venn diagrams in Figure 33. A stable set of a markings is shown in blue. Here, no step leaves the blue set, which is equivalent to the definition of stability. The green set shows the reachable markings, the red set shows the reachable and invalid markings. The initial marking is depicted as a black dot. Intuitively, the lemma states that exactly one of the both Venn diagrams is true: The first case is shown in Figure 33a: The initial marking is in the blue set. Then, all reachable markings are in the blue set. The second case is shown in Figure 33b: The initial marking is in the red set. Then, trivially not all reachable markings are in the blue set.

4.2.1 Non-Preserving Steps

In the last section, we defined stability of algebraic Petri nets. In this section, we shift the perspective slightly and formulate an equivalent definition based on the steps of an algebraic Petri net. Stability and validity of a set of markings may be expressed in terms of *preserving* and *non-preserving steps*.



(a) A stable, valid set of markings.



(b) A stable, invalid set of markings.

Figure 33.: Lemma 46 visualized as two Venn diagrams. The blue set depicts a stable set of markings, the green set depicts the valid reachable markings, the red set depicts the invalid unreachable markings, and the dot depicts the initial marking.

Definition 47 ((Non-)Preserving Step)
Let $(P, F, \equiv, T)$ be an algebraic Petri net structure. Let $\phi \subseteq \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$ be a set of markings. Let $(m, m') \in \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P \times \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$ be a step.
Then, (ψ, ψ') is ϕ-*preserving*, if $m \in \phi$ implies $m' \in \phi$.
Then, (ψ, ψ') is ϕ-*non-preserving*, if $m \in \phi$ and $m' \notin \phi$.

With the following corollary we see that the notion non-preserving steps induces an alternative characterization of stability: A set of markings is stable if and only if all steps are preserving.

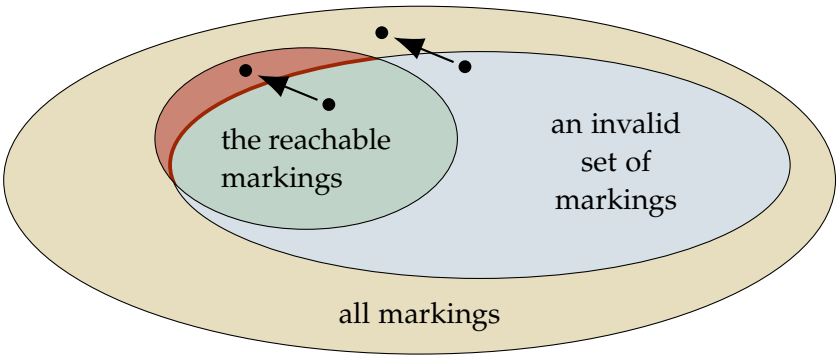


Figure 34.: Non-preserving steps of an invalid set of markings as Venn diagram with two non-preserving steps.

Corollary 48 (Stability and Preservation)
Let $(P, F, \equiv, T)$ be an algebraic Petri net structure. Let $\phi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P}$.

Then, the following statements are equivalent:

1. ϕ is stable in $(P, F, \equiv, T)$.
2. All steps $(m, m') \in \mathcal{R}(T)^{\uparrow}$ are ϕ-preserving.

In Figure 32, the set $\{m^{32.2}\}$ is not stable, because there exists the non-preserving step $(m^{32.2}, m^{32.1}) \in \mathcal{R}(a)^{\uparrow}$. On the contrary, the set $\{m^{32.1}, m^{32.2}\}$ is stable, as the step $(m^{32.2}, m^{32.1})$ is $\{m^{32.1}, m^{32.2}\}$-preserving. Hence, a non-preserving step is a counter-example for stability.

If the source marking of a non-preserving step is reachable, then the non-preserving step is moreover a counterexample for validity.

Corollary 49 (A Non-Preserving Step Disproves Validity)
Let $(P, F, \equiv, T)$ be an algebraic Petri net structure. Let $\phi \subseteq \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P}$ and $m^0 \in \phi$.

Then, the following statements are equivalent:

1. ϕ is invalid.
2. There exist a step $(m, m') \in \mathcal{R}(T)^{\uparrow}$ such that (m, m') is non-preserving and $m \in \textsc{Reach}_T(m^0)$.

Using Definition 47 and Corollaries 48 and 49, we now have a formalization of the Venn diagram motivated in Chapter 1, which is restated in Figure 34. The blue set depicts an invalid subset of the set of all markings. The union of the green and the red set depicts the set of all reachable markings. The arrows with dots depict two non-

preserving step: The left starts from a reachable marking, the right starts from a non-reachable marking. The right step proves that the blue set is not stable. The left step proves additionally that the blue set is not valid.

4.2.2 Using Stability for Analysis and Verification

Given an algebraic Petri net N and an algebraic equation or inequality I, the problem of validity of I in N is undecidable, as shown in Theorem 44. However, if I is stable, then deciding validity of I reduces to checking if the initial marking satisfies I, as shown in Lemma 46. Hence, assuming that stability and satisfaction are tractable, they together induce a sufficient and tractable criterion, which can be used for computational analysis and verification. Thus, stability can be used as an incomplete verification technique.

Another, more involved approach for verification is the following: To verify validity of I in N, we use a second algebraic equation or inequality I', such that $\mathbb{L}(I') \subseteq \mathbb{L}(I)$ and I' is stable and valid in the initial marking. Then, we deduce by Lemma 46 that I' is valid. Moreover, as $\mathbb{L}(I') \subseteq \mathbb{L}(I)$, we deduce that I is also valid. A stricter and valid algebraic equation or inequality implies validity of the original equation or inequality. The approach is sketched as a Venn diagram in Figure 35. Again, the set of all markings is sand-colored, and the set of reachable markings is green. Moreover, the Venn diagram shows two valid sets of markings: The smaller, dark blue set, is valid and stable. The larger, light blue set is also valid, as by transitivity this set subsumes the set of reachable markings. We deduce the validity of the larger set of markings solely from the fact that the set subsumes a smaller valid set. The challenge in this approach is to find a stable, valid and stricter algebraic equation or inequality I' for the given algebraic equation or inequality I.

Summarizing, given an algebraic equation or inequality I and its set of non-preserving steps $M(I)$, we can reduce validity in all reachable markings to validity in the initial marking, if $M(I) = \emptyset$ by Lemma 46 and Corollary 48. Otherwise, a non-preserving step $(m, m') \in M(I)$ is a counter-example for stability and a potential counter-example for validity. Thus, given a subset of markings, the set of non-preserving steps is crucial for the analysis and verification. In Part II, we will show that a non-preserving step is computable for a given algebraic equation or inequality, if the underlying data scheme is compact.

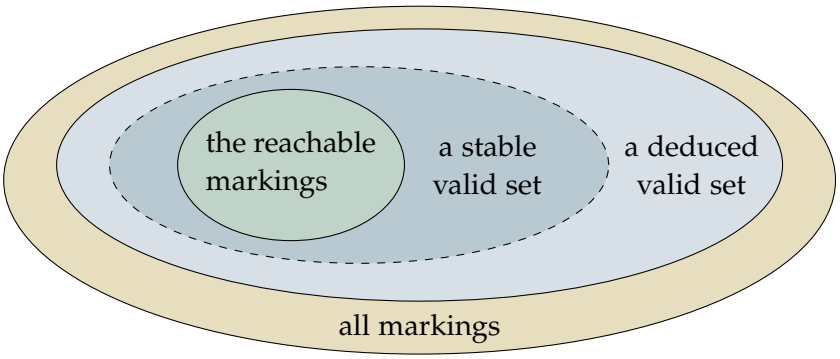


Figure 35.: Deducing validity.

4.3 DISCUSSION

In this chapter, we studied analysis and verification of an algebraic equation or inequality in an algebraic Petri net. We showed that validity of an algebraic equation and inequality is undecidable, even if a compact data scheme is considered. As a second step, we recalled stable sets of markings and showed how stability can be used to analyze and verify algebraic equations and inequalities. In the following sections we discuss the undecidability result and the concept of stability in algebraic Petri nets with respect to related work.

4.3.1 Undecidability Results

In section Section 4.1, we have shown that place emptiness is undecidable in Lemma 41. We have published the undecidability proof in [TS16b]. The technical report with full proofs can be found in [TS16c]. For the publications [TS16b; TS16c], the author of this thesis contributed the idea for the proof. Carving out the main lemmas and organizing the proof was done in cooperation with the second author. The introduction and conclusion was written by the second author. There, we restricted the proof to algebraic equations. However, the generalization for algebraic inequalities follows immediately. The technique to reduce Minsky machines is a common technique in the fields of Petri nets to encode a Turing-complete formalism that implies undecidability [Pop13; RS09].

It is well-known that verification problems become undecidable when a combination of data and processes is considered [Bag+13; Wag15; MR16a; MR17; Köh07; De +17]. Typically, analysis opportunities are obtained by restricting the models and bounding the data model. For example in [Bag+13], the number of distinguishable entities is bounded in each state. In [Wag15], the system is expected to

be acyclic and entities to be bounded. In [De +17], the entities are bounded.

COVERABILITY. *Coverability* is defined for infinite state systems with an order on states: A state is coverable, if there exists a reachable state that is greater with respect to that order. In Petri nets, markings are ordered by $\geqslant$. We observe that undecidability of place emptiness also implies undecidability of coverability in algebraic Petri nets: The problem whether a specific term is assigned to a place by a reachable marking can be reduced to the problem whether a reachable marking assigns at least a specific term to that place. As coverability is not at the core of our discussion, we leave a full proof for future work. However, we observe that algebraic Petri nets fall not into the category of *well-structured transition systems* [FS01; Abd+96], which have been used for verification of infinite state systems [MMW13]. In well-structured transitions systems, reachability is undecidable, but coverability is decidable [Abd+96; GRB04]. Moreover, in [Las16], an overview on the expressiveness regarding decidability of coverability is given. In that context, algebraic Petri nets are a very expressive formalism from the complexity point-of-view, even when considering a compact data scheme.

FINITE DATA SCHEME. For many modeling purposes, it is convenient to model an infinite set of entities. However, a data scheme may also model finitely many entities. In this case, the undecidability of place emptiness does not carry over. The proof of Lemma 41 exploits that the data scheme $(\mathsf{FN}, \equiv_\emptyset)$ induces an infinite number of terms. Considering a finite data scheme, an algebraic Petri net may be unfolded into a elementary Petri net as follows: The places of the elementary Petri net are the Cartesian product of places and terms. Transition are unfolded according to all possible firing modes. Then, analysis and verification techniques for elementary Petri nets with indistinguishable tokens may be applied [Rei13]. However, in this thesis we focus on algebraic Petri nets over a data scheme that induces an infinite number of terms.

4.3.2 Stability

The concept of stability is an established and long-known technique [Flo67; Hoa72; Dij80] to prove correctness of a system.

Whereas for a long time the finding of stable properties has been left to a human [Gri81], in the recent years, several techniques for inferring stable properties automatically have been proposed [FMV14].

However, in our work we do not study how to infer stable algebraic equations and inequalities from a given algebraic Petri net. This field of research is left for future work. We focus on analysis of a given algebraic equation or inequality.

4.3.3 Stability in Petri Nets

In the last years, stable equations have been studied for the verification of distributed systems in the context of P-*invariants* [Vau85; Vau87; MV86; GL79; GLT79; Rei13; Rei91; Rei98; Mur89; Jen81b; Jen81a]. Each P-invariant yields a stable equation. However, not every stable equation is a P-invariant. In [TS16b; TS16c], we characterized the class of stable homogeneous algebraic equation, subsuming the class of P-invariants.

Regarding inequalities, the class of *traps* and *co-traps* yield stable inequalities [Hac72; HGD08; Mur89; Rei13]. Again, traps and co-traps yield stable inequalities, but not every stable inequality is a trap or co-trap. Moreover, the work on traps and co-traps has been studied mostly for indistinguishable tokens, only in [Sch97; KV98] data modeling is considered.

We generalized these results and published a full characterization of inequalities in [TS15; TS16a] for Petri nets with indistinguishable tokens.

In Part II, we generalize our results such that we provide a tractable characterization of all stable algebraic equations and inequalities of algebraic Petri nets over a compact data scheme.

4.3.4 Verification of Distributed Systems

The analysis and verification of Petri nets for behavioral properties in distributed systems has been studied in terms of service automata and so-called service nets [GMW12]. An important result is the characterization of all environments of a service net such that the overall system terminates weakly [LMW07]. Based thereon, more involved questions such as adaptability of two service nets [GMW12] or cost-optimal communicating service nets [ST13] have been studied. However, these works typically abstracts from data and handle messages as non-distinguishable entities, i.e. as tokens.

4.3.5 Verification of Data-Aware Systems

In [MR15; MR16b] the authors have shown a method for verification of temporal formulas under the assumption of run-boundedness for Petri nets using pure names as tokens. More specifically, the results from data-centric dynamic systems [Bag+13] are applied.

In [MNS14; SMN16] data-aware business processes are studied. Data-aware business processes are finite-state transition systems with an underlying database. Datalog [Dan+97] queries access and manipulate the underlying relational database. Verification methods were investigated, for instance for the question of query stability [SMN16]. However, data-aware business processes consider a monolithic process. They do not incorporate distributed systems or communication between different agents.

For colored Petri nets, model checking, testing has established for analysis [JK09]. The approach has been adapted for algebraic Petri nets [Kha13; KR13; KG14b; KG14a; Kha15]. However, model checking and testing are incomplete verification methods and can only show violations. Proving correctness is impossible.

5 ILLUSTRATIVE EXAMPLE: PURCHASE ORDER

In this section, we illustrate preservation of data integrity using the example of a purchase order system. The purchase order system is distributed among several participants and locations. In Section 5.1, we model the system as the algebraic Petri net N_{PO}. In Section 5.2, we specify data integrity with the algebraic inequality $\mathsf{I}^1_{\mathsf{PO}}$. We show that $\mathsf{I}^1_{\mathsf{PO}}$ is stable and valid in N_{PO}. In Section 5.3, we study the algebraic inequality $\mathsf{I}^2_{\mathsf{PO}}$. We show that $\mathsf{I}^2_{\mathsf{PO}}$ is not stable in N_{PO}. However, we deduce validity of $\mathsf{I}^2_{\mathsf{PO}}$ using $\mathsf{I}^1_{\mathsf{PO}}$. In Section 5.4, we study the algebraic equation $\mathsf{I}^3_{\mathsf{PO}}$. We show that $\mathsf{I}^3_{\mathsf{PO}}$ is neither stable nor valid. In Section 5.5, we discuss peculiarities of the illustrative example. All proofs in this section are informal. We use intuitive reasoning rather than formal proofs. We emphasize the role of non-preserving steps and how they are used for the analysis and verification of data-aware distributed systems.

5.1 MODELING THE PURCHASE ORDER SYSTEM

In this section, we describe a distributed purchase order system and model the system as an algebraic Petri net. The example is inspired by the order fulfillment process from [Dum+13]. However, our model is not motivated by a case study and serves solely for illustration. In our example, we model multiple customers and multiple assets. The production, order and delivery of an asset is decoupled from the offer and pick-up.

In the modeled system, every asset is stored in storage and concurrently offered and sold. This way, the order and pick-up of an asset is decoupled. An offer is a reference to an actual asset, which is stored in the storage. Each customer that accepts an offer gets a voucher for the asset. However, the customer retrieves the purchased asset as soon as he picks it up from the storage with the voucher.

As offers and vouchers are referential data entities, we study referential integrity. Intuitively, referenced assets should be available in the storage, as we will study in Sections 5.2 to 5.4.

places P_{PO}:	Fresh, Storage, Vitrine, Shop, Wallet
function symbols F_{PO}:	👤 : *Customers*,
	next : *Customers* $\to$ *Customers*,
	🚲 : *Assets*,
	next : *Assets* $\to$ *Assets*,
	💶 $\to$ *Prices*,
	succ : *Prices* $\to$ *Prices*,
	offer : *Assets* $\times$ *Prices* $\to$ *Offers*,
	voucher : *Assets* $\times$ *Customers* $\to$ *Vouchers*,
	asset : *Offer* $\to$ *Assets*,
	asset : *Vouchers* $\to$ *Assets*,
axioms A_{PO}:	asset(offer(**g**,**p**)) $\doteq$ **g**
	asset(voucher(**g**,**c**)) $\doteq$ **g**
$\text{SORTS}(F_{PO})$:	*Assets*, *Prices*, *Customers*, *Offers*, *Vouchers*
variables:	$\mathbf{a} \in \mathbb{V}_{Assets}$
	$\mathbf{c} \in \mathbb{V}_{Customers}$
	$\mathbf{p} \in \mathbb{V}_{Prices}$
	$\mathbf{o} \in \mathbb{V}_{Offers}$
	$\mathbf{v} \in \mathbb{V}_{Vouchers}$

Figure 36.: The distributed data scheme $(P_{PO}, F_{PO}, \equiv_{A_{PO}})$ modeling the entities of the purchase order system.

5.1.1 Compact Data Scheme

In the following paragraphs, we describe a data scheme to model the entities of the purchase order system. The resulting data scheme $(P_{PO}, F_{PO}, \equiv_{A_{PO}})$ is shown in Figure 36. The compact data scheme contains five sorts. Each sort is described in one of the following paragraphs.

CUSTOMERS. For a purchase order, a company typically stores different information about a customer, e.g. the name, delivery address or date of birth. However, the behavior of the system does not depend on these values. Hence, we abstract from them. We assume an unbounded number of customers. To model an infinite set of customers using a data scheme, we model an order on customers: Starting from

an initial customer 👤, we may enumerate the following customers: next(👤), next(next(👤)), ... In $(F_{PO}, \equiv_{A_{PO}})$, we use the sort *Customers* and two function symbols: 👤 $\rightarrow$ *Customers* and next : *Customers* $\rightarrow$ *Customers*. Accordingly, two customers described by terms with the symbols next and 👤 are equivalent if and only if the terms are equal.

ASSETS. We model assets similar to customers and abstract from further attributes such as age or weight. Assuming an unbounded number of assets, we model a "first" asset and respective following assets. The model for asset is built on the sort *Assets*. We use two function symbols 🚲 and new to indicate the "first" asset 🚲 and all following assets by new(🚲), new(new(🚲)), etc.

PRICES. For the model of prices, we abstract from the currency. We assume a lowest price 💶, and model a higher successor price for every existing price. We model prices using the sort *Prices*. We use two function symbols 💶 and suc. Hence, the terms modeling prices are 💶, suc(💶), etc.

OFFERS. An offer in the purchase order system is a tuple of an asset and a price. Hence, we use the function symbol offer : *Assets* $\times$ *Prices* $\rightarrow$ *Offers* to model offers. We observe that the function symbol offer is the only function symbol that may create a term of the sort *Offers*. Hence, every term of sort *Offers* is generated using this function symbol. Additionally, the model contains the function symbol asset : *Offers* $\rightarrow$ *Assets*. This function symbol is the projection of an offer to its asset. We use the construction of asset data scheme as defined in Definition 6 (Page 37). The axiom asset(offer(**go**, **pr**)) $\doteq$ **go** induces the semantic of the projection. For example, the term offer(🚲,suc(0)) satisfies the following equivalence:

$$\text{asset}(\text{offer}(🚲,💶)) \equiv_{A_{PO}} 🚲$$

VOUCHER. A voucher is an entity that combines a customer with a asset. A customer may use a voucher to pick up an asset. We model a voucher as a tuple of an asset and a customer, similarly to offers. We use the function symbol voucher : *Assets* $\times$ *Customers* $\rightarrow$ *Vouchers*. For vouchers we model the projection function asset. For example a customer 👤 may own a voucher for a 🚲 using the voucher voucher(🚲,👤). Then, we have the following equivalence induced by A_{PO}:

$$\text{asset}(\text{voucher}(🚲,👤)) \equiv_{A_{PO}} 🚲$$

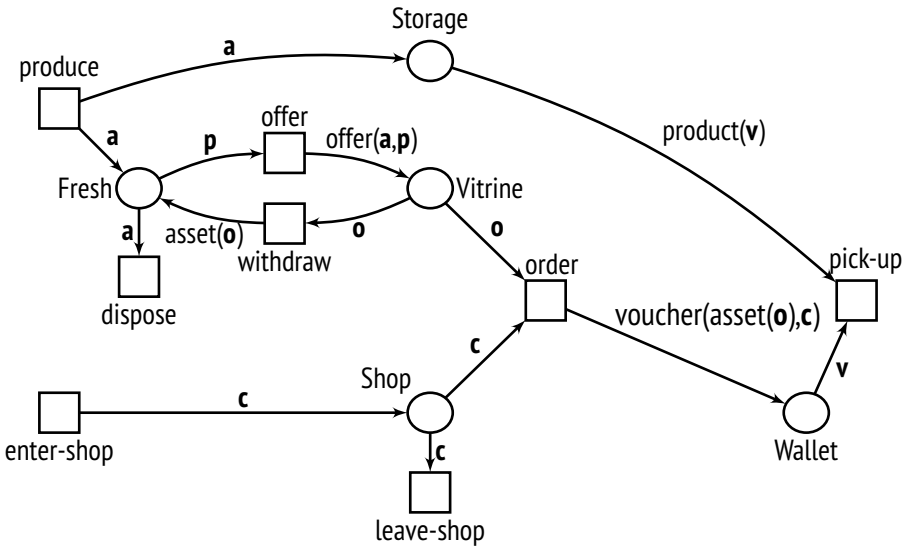


Figure 37.: A purchase order system modeled as algebraic Petri net N_{PO}

COMPACTNESS. In this paragraph, we sketch a proof to show that $(F_{PO}, \equiv_{A_{PO}})$ is compact. Assume three disjoint sets of function symbols for customers, prices and orders. Then, each set with the identity as congruence is a compact data scheme, as they are Herbrand structures. The data scheme shown in Figure 36 is constructed using the asset of compact data schemes with projection as defined in Definition 6 (Page 37) twice. The compactness is implied by Lemma 7 (Page 38). Hence, we deduce that the resulting data scheme is a compact data scheme.

5.1.2 Algebraic Petri Net

The algebraic Petri net N_{PO} shown in Figure 37 models the purchase order system. The underlying compact data scheme is shown in Figure 36. The distributed data scheme contains five places in P_{PO}: Fresh, Shop, Vitrine, Wallet, and Storage with $\text{SORT}(\text{Fresh}) = \text{SORT}(\text{Storage}) = \textit{Assets}$, $\text{SORT}(\text{Shop}) = \textit{Customers}$, $\text{SORT}(\text{Vitrine}) = \textit{Offers}$, and $\text{SORT}(\text{Wallet}) = \textit{Vouchers}$. First, we describe the behavior of an asset. Secondly, we describe the behavior of a customer.

When an asset is produced, it is stored in the storage. The same asset may be produced more than once. The transition produce models the production. produce has no precondition. The place Fresh is not modeling a physical location, but rather a logical condition of assets. Accordingly, produce has two post places: Fresh and Storage. The transition offer models the offering of fresh assets: An asset is consumed

from the place Fresh and an offer is produced on the place Vitrine. Here, the variable **p** is not used in the precondition. Hence, the price may obtain any value in the step. An offer may be withdrawn. The corresponding asset may be offered again for a different price. The transition withdraw models withdrawals: An offer is consumed from place Vitrine and the according asset is stored at the place Fresh. Then, the asset is either disposed or offered again. Disposals are modeled by the dispose transition. When a customer orders an offer from the vitrine, he gets a voucher in his wallet. Orders are modeled by the transition Order. order consumes an order from the Vitrine and a customer from the place Shop. Moreover, order produces a voucher on the place Wallet. Here, the place Wallet does not model a single location. The wallet is different for each customer. However, in N_{PO}, we model the wallet as a logical condition using one place. Finally, a customer may pickup an asset from the storage with an according voucher. The pick-up of assets is modeled by the transition pick-up, that consumes a voucher from Wallet and the corresponding asset from Storage.

A customer is not unique and may be in the shop as a duplicate. Any customer can enter the shop, which is modeled by the transition enter-shop and the place Shop. The number of customers in the shop is not bounded. Hence, the precondition enter-shop^- is 0 for all places. A customer may also leave the shop without ordering an asset, which is modeled by the transition leave-shop. leave-shop consumes a customer from the place Shop and the post condition is zero for all places. A customer may accept an offer and order an asset, which is modeled by the transition order. Then, the customer leaves the shop, but retrieves a voucher in his wallet. This is modeled by producing the respective voucher term on Wallet. Using a voucher from his wallet, a customer may pick up the ordered asset from the storage, as modeled by the transition pick up. While picking up, a customer leaves the purchase order system. The post condition of pick-up is zero for all places.

Initially, the storage is empty, no customer is in the shop, no assets are offered or ordered. Accordingly, the initial marking is empty for all places.

5.2 A VALID, STABLE INEQUALITY

In this section, we will specify data integrity in the purchase order system by the algebraic inequality I^1_{PO}. We will show that the inequality is stable and valid in N_{PO}.

algebraic equation or inequality		stable?	valid?
I^1_{PO}:	**Storage** − **Fresh** − asset(**Vitrine**) − asset(**Wallet**) $\dot{\geq}_{A_{PO}}$ [ ]	✓	✓
I^2_{PO}:	**Storage** − asset(**Wallet**) $\dot{\geq}_{A_{PO}}$ [ ]	×	✓
I^3_{PO}:	**Storage** − **Fresh** − asset(**Vitrine**) − asset(**Wallet**) $\doteq_{A_{PO}}$ [ ]	×	×

Figure 38.: Two algebraic inequalities and one algebraic equation of the algebraic Petri net N_{PO}.

INTEGRITY OF REFERENCED ASSETS. In N_{PO}, every asset is stored in storage and concurrently offered and sold. This way, the pick-up of an asset and the order are decoupled. Each order and each voucher contains a reference to an asset that is stored in storage. Therefore, the requirement of data integrity is with respect to references of assets. Intuitively, it is expected from the system that the sum of all assets that refer to an asset in the storage is not greater than the assets in the storage. More specifically, it is expected that there exists a distinct asset in the storage for every reference. For example, an asset which was already ordered by a customer should not be offered. Therefore, it is expected that offered assets and ordered assets are distinct. This intuition of data integrity is formalized with the algebraic inequality I^1_{PO} as shown in Figure 38. Here, the integer coefficient is 1 for the place Storage. For the places Fresh, Vitrine, and Wallet the integer coefficient is -1. All other places are not taken into account and their integer coefficient is 0. Hence, the assets in Storage are considered and the assets from Fresh, Vitrine, and Wallet are subtracted. Since in the vitrine offers are stored, we apply the term induced function asset(**Vitrine**), which projects each offer to its assets. This equivalence is induced by the first axiom of A_{PO} as shown in Figure 36. The same is achieved using the function induced by the term asset(**Wallet**) for the place Wallet.

As the right-hand side of I^1_{PO}, we have the empty bag [ ]. Accordingly, if the difference is semi-positive, there exist a distinct asset in the storage for every referenced asset.

STABILITY. In this paragraph, we show that the algebraic inequality I^1_{PO} is stable in N_{PO}. We will not show a full formal proof, but give an informal explanation. We consider an arbitrary step (m, m') of N_{PO}, such that m satisfies I^1_{PO}. It suffices to show that m' also satisfies I^1_{PO}. We distinguish the cases by the transition which induced the step (m, m').

1^{st} case: leave-shop.
In this case, the places Storage, Fresh, Vitrine and Wallet are not af-

fected by the transition. Hence, we deduce that $m_{\text{Storage}} = m'_{\text{Storage}}$, $m_{\text{Fresh}} = m'_{\text{Fresh}}$, $m_{\text{Vitrine}} = m'_{\text{Vitrine}}$, and $m_{\text{Wallet}} = m'_{\text{Wallet}}$. Thus, the referenced assets and the stored assets do not change in the step and I^1_{P0} is satisfied in m'.

2^{nd} case: produce.
In this case, a new asset is produced and put on Fresh and Storage. As the inequality is satisfied in m, it is sufficient to consider this newly produced asset. As the coefficient is 1 for Storage, and -1 for Fresh, the newly added assets cancels out.

3^{rd} case: offer.
A reference is "moved" from Fresh to Vitrine. Hence, the number of references does not change and the assets in Storage are not affected. We deduce that the inequality is satisfied in m'.

4^{th} case: withdraw.
Again, a referenced asset is moved by the transition. As both places have the same coefficient -1, the result of the abstraction query of I^1_{P0} applied to m yields the same result as applied to m'.

5^{th} case: order.
The transition moves a reference from Vitrine to Wallet. Here, the reference is extracted from the offer, using the function asset(**o**), where **o** is a variable over offers. The reference is then stored in a voucher. The abstraction query of I^1_{P0} considers the references extracted from vouchers from the place Wallet and extracted from offers from the place Vitrine equally, as both places have the same coefficient -1. Hence, the referenced assets do not change and the abstraction query of I^1_{P0} yields the same result for m and m'.

6^{th} case: pickup.
Here, an asset is consumed from Storage and a voucher is consumed from Wallet. The voucher references the asset that is stored in Storage. Hence, a reference and a stored asset is removed. Thus, the result of the abstraction query is the same for m and m'. We deduce: $m' \in \mathbb{L}(I^1_{P0})$.

7^{th} case: dispose.
The transition consumes a reference to an asset from the place Fresh. As the coefficient of Fresh is -1, the result of the abstraction query of m' is greater equal the result of m. And hence, m' satisfies I_{P0}.

Thus, every step of N_{P0} preserves I^1_{P0} and accordingly the set of non-preserving steps of I^1_{P0} is empty.

VALIDITY. In the rest of this section, we show that I^1_{PO} is valid. By Lemma 46 (Page 91) it is sufficient to show that the initial marking satisfies I^1_{PO}. As the initial marking is empty for all places, the result of the abstraction query is zero. Thus, the initial marking satisfies I^1_{PO}. Furthermore, as I^1_{PO} is stable, we derive that I^1_{PO} is valid. Thus, the model N_{PO} satisfies data integrity with respect to this specification.

As we have proven validity of I^1_{PO} in N_{PO}, we have proven that all reachable markings satisfy the specification of data integrity. Hence, each implementation of the model satisfies data integrity with respect to referenced assets.

5.3 A VALID, UNSTABLE INEQUALITY

In this section, we study a different specification of data integrity in the purchase order system. We formalize this specification using the algebraic inequality I^2_{PO} as shown in Figure 38. We will show that I^2_{PO} is not stable, but valid.

ASSETS REFERENCED BY VOUCHERS. In the purchase order system, a customer may have a voucher of an asset in his wallet. For each voucher in the wallet it is expected that a distinct asset in the storage exists. All other references of assets are not taken into account. This specification of data integrity is similar to the specification by I^2_{PO} discussed in the previous section. But in contrast to I^1_{PO}, only the references in the Wallet are considered.

The specification described above is formalized by the algebraic inequality I^2_{PO}. In the abstraction query only two coefficients are unequal zero: For Storage it is 1, and for Wallet it is -1. The abstraction query of I^2_{PO} considers the difference between the assets in Storage and the referenced assets in Wallet. For the place Wallet, the term asset(**Wallet**) induces a function on terms. The function projects vouchers to the referenced asset. Assume a marking m with more assets in the Storage than references by vouchers in Wallet. Then, the abstraction query applied to m is greater or equal to zero and I^2_{PO} is satisfied by m. Otherwise, the result is not greater equal zero and there are assets referenced by vouchers, which cannot be found in the storage.

INSTABILITY OF I^2_{PO}. In this paragraph, we show that I^2_{PO} is not stable. There exists a non-preserving step as shown in Figure 39. Intuitively, a single asset may be offered and ordered at the same time. Then, a second order creates a second voucher for the same asset. While the source marking m^1 is a solution of I^2_{PO}, the target marking m^2 is

p:	Fresh	Storage	Vitrine	Shop	Wallet
m_p^1	[ ]	[🚲]	[offer(🚲, 💶)]	[next(👤)]	[voucher(🚲, 👤)]
m_p^2	[ ]	[🚲]	[ ]	[ ]	[voucher(🚲, 👤) , voucher(🚲, next(👤))]

Figure 39.: The step (m^1, m^2) of N_{P0}, which does not preserve I_{P0}^2.

not a solution of I_{P0}^2. By Corollary 48 (Page 93), we derive that I_{P0}^2 is not stable in N_{P0} as there exists a non-preserving step. However, the marking m^2 is not reachable.

VALIDITY OF I_{P0}^2. In this paragraph, we deduce validity of I_{P0}^2. To do this, we use the valid inequality I_{P0}^1. In the following, we observe that every solution of I_{P0}^1 is also a solution of I_{P0}^2: For all places p it holds that the integer coefficient of I_{P0}^1 is smaller or less the integer coefficient of I_{P0}^2. Moreover, the term coefficients are equal. Accordingly, the result of applying the abstraction query of I_{P0}^1 is smaller or less than applying the abstraction query of I_{P0}^2. Thus, each solution of I_{P0}^1 is a solution of I_{P0}^2. As I_{P0}^1 is valid, we derive:

$$\textsc{Reach}_T(m^0) \subseteq \mathbb{L}(I_{P0}^1) \subseteq \mathbb{L}(I_{P0}^2)$$

Thus, we deduce validity of I_{P0}^2.

For this example, we were able to deduce validity of I_{P0}^2 using the valid and stable algebraic inequality I_{P0}^1. However, in general it is undecidable whether an algebraic inequality is valid as shown in Theorem 44. It is a non-trivial task to find an according valid algebraic inequality which implies validity of a given algebraic inequality. Therefore, we will not try to find a method to derive such an inequality in this thesis.

5.4 AN INVALID EQUATION

In this section, we study a third specification of data integrity in the purchase order system. We formalize this specification using the algebraic equation I_{P0}^3 as shown in Figure 36. We will show that I_{P0}^3 is neither stable nor valid.

ALL REFERENCES REFLECT THE STORAGE (INVALID). The third specification of data integrity has the following semantics: Do all references to products in the storage precisely reflect the contents of prod-

p:	Fresh	Storage	Vitrine	Shop	Wallet	$\in \mathbb{L}(I^3_{PO})$?
m^0_p	[ ]	[ ]	[ ]	[ ]	[ ]	✓
m^3_p	[🚲]	[🚲]	[ ]	[ ]	[ ]	✓
m^4_p	[ ]	[🚲]	[ ]	[ ]	[ ]	×

Figure 40.: The three markings m^0, m^3, and m^4, where (m^0, m^3) and (m^3, m^4) are steps.

ucts in the storage? This specification is formalized as I^3_{PO} as shown in Figure 38. The abstraction query of I^3_{PO} is the same as for I^1_{PO}. However, I^3_{PO} is an algebraic equation and I^1_{PO} is an algebraic inequality. We will show that I^3_{PO} is not valid in N_{PO}. As discussed in Section 5.2, the abstraction query considers the difference between all products in the storage and all referenced products in the system. If the difference is zero, all references sum up to the bag of products stored in the storage.

INSTABILITY OF I^3_{PO}. In this paragraph, we show that I^3_{PO} is not stable. By Corollary 48 (Page 93), it is sufficient to show that there exists a non-preserving step. The non-preserving step (m^3, m^4) is shown in Figure 40. In m^3, the product 🚲 is in Storage and referenced on Fresh. Hence, the difference is $[🚲] - [🚲] = 0$ and $m^3 \in \mathbb{L}(I^3_{PO})$. Then, (m^3, m^4) is step of the transition withdraw. In m^4, the withdrawn product is not fresh, but still in the storage. Accordingly, we have $(\ell^1, o^1) \odot m^4 = [🚲]$. Thus, m^4 is not a satisfying marking of I^3_{PO}. As (m^3, m^4) is a non-preserving step, we deduce that I^3_{PO} is not stable.

INVALIDITY OF I^3_{PO}. In this paragraph, we show that I^3_{PO} is not valid. It suffices to show that a non-satisfying marking is reachable. As shown in the previous paragraph, m^4 from Figure 40 is a non-satisfying marking. We observe that (m^0, m^3) is a step of the transition produce. Thus, (m^0, m^3) and (m^3, m^4) are steps of N_{PO} and m^4 is reachable. Hence, I^4_{PO} is not valid.

In general, validity is undecidable, but non-preserving steps are computable. In this example, a non-preserving step produces a counterexample for stability. We extended that counterexample for stability to a counterexample for validity. Thus, in this example the non-preserving step served as an indicator for the analysis of the purchase order system. However, it is not decidable whether the markings of a non-preserving step are reachable.

5.5 DISCUSSION

In this chapter we showed the example of the purchase order system modeled as the algebraic Petri net N_{PO} over the compact data scheme $(F_{PO}, \equiv_{A_{PO}})$. We specified data integrity using the valid algebraic inequalities I^1_{PO} and I^2_{PO}, and the invalid algebraic equation I^3_{PO}.

We have proven validity of I^1_{PO} by showing that all steps of N_{PO} are I^1_{PO}-preserving. Equivalently, the set of non-preserving steps is empty. For I^2_{PO}, the set of non-preserving steps is not empty. Hence, I^2_{PO} is not stable. Nevertheless, we have proven validity of I^2_{PO}. We used the validity of I^1_{PO} to deduce validity of I^2_{PO}. Hence, all non-preserving steps of I^2_{PO} are not reachable. We have proven invalidity of I^3_{PO}. Here, a non-preserving step to a reachable marking was used as a counterexample for stability and validity.

In all three sections, we studied the role of non-preserving steps and used them for the analysis and verification of algebraic equations and inequalities in an algebraic Petri net. Thus, a finite representation of all non-preserving steps is a crucial tool for analysis. We will show computability of the set of non-preserving steps in Part II. We study a prototype implementation for the computation of non-preserving steps in Chapter 11. We will model N_{PO} using the software CPNTools [JK09] and show how our prototype supports the analysis of the shown algebraic equation and inequalities.

5.5.1 Restrictions of the Example

In our illustrative example, we have not used the full expressive power of algebraic Petri nets and algebraic equations and inequalities. The shown example is restrictive with respect to the following four points.

SIMPLE TRANSITIONS. We observe that every arc inscription of N_{PO} is a monomial of the form $1 \cdot \theta$ for some term θ or 0. However, in general an arc inscription may be an arbitrary bag of terms. The example demonstrates that monomial arc inscriptions are sufficient to model interesting problems.

SIMPLE COEFFICIENTS. In the abstraction queries ℓ^1 and ℓ^2, all coefficients are $-1, 0$, or 1. In general, the coefficients may be any integer. In the example, these coefficients are sufficient to express referential integrity. Considering different integers enables the possibility to quantify references. An example would be the requirement

that for every order there be at least three assets in storage. However, we left this possibility out of the scope of the illustrative example.

HOMOGENEITY. In the algebraic inequalities $\mathsf{I}^1_{\mathsf{PO}}$ and $\mathsf{I}^2_{\mathsf{PO}}$, and in the algebraic equation $\mathsf{I}^3_{\mathsf{PO}}$, the right-hand side is the empty bag $[\,]$. In general, the right-hand side may be any polynomial over ground terms. For the example, these coefficients are sufficient to express referential integrity.

COMPACT DATA SCHEME. The algebraic Petri net N_{PO} uses a compact data scheme. The example illustrates that it is possible to model entities of a data-aware system. As we will show in Part II, the computation of non-preserving steps is decidable, if the data scheme of the algebraic Petri net is compact. In general, many analysis properties become undecidable considering arbitrary data schemes.

Part II.

Computing Non-Preserving Steps

In part II, we show that non-preserving steps are computable, if the underlying data scheme is compact. We give an overview of the proof in Chapter 6, reduce from general transitions to *simple transitions* in Chapter 7, study solutions of *Diophantine equations* and *inequalities* in Chapter 8, show computability of a *linear representation* of the set of solutions of an *algebraic equation* or *inequality* in Chapter 9, and show computability of steps from a linear represented set of markings in Chapter 10.

6 OVERVIEW

In this chapter, we give an overview of Part II. The main result of Part II is Theorem 103 (Page 183): The computability of non-preserving steps if the underlying data scheme is compact.

We give an overview how de derive the proof of Theorem 103. The crucial part of the proof is the computability of a *linear representation*, which we define in Section 6.1.

In Section 6.2, we summarize the proof of computation by dividing it in four computational tasks. Each of these tasks is studied in detail in one of the following chapters.

6.1 LINEAR REPRESENTATION

In this section, we introduce *linear representations*. The definition is crucial for the computability, as a linear representation of all solutions of an algebraic equation or inequality is computable, if the underlying data scheme is compact, as shown in Chapter 9. In linear algebra, the set of solutions of an equation can be described by a translation vector and a set of basis vectors. Then, the translation vector summed with a linear combination of basis vectors yields a solution. We follow a similar approach, but consider markings instead of vectors. Moreover, both the set of "translation markings" and the set of "basis markings" may be infinite. Hence, we represent them by finite sets of parameterized markings. Two finite sets of parameterized markings induce their *linear realizations* by summing a realization from one set with a sum of realizations from the other set.

> **Definition 50** (Linear Realization, Linear Representation)
> Let $M \subseteq \mathbb{N}\langle {\emptyset \atop F} \rangle^P$. Let $S, Z \subset \mathbb{N}\langle {V \atop F} \rangle^P$ be finite sets of parameterized markings. Then, we define the *linear realizations of* S, Z, denoted by $\textsc{Lin}(S, Z) \subseteq \mathbb{N}\langle {\emptyset \atop F} \rangle^P$ as follows:
>
> $$\textsc{Lin}(S, Z) = \left\{ s + z^1 + \cdots + z^n \mid s \in \mathcal{R}(S) \text{ and } z1, \ldots, z^n \in \mathcal{R}(Z) \right\}$$
>
> If $\textsc{Lin}(S, Z) = M$, we call S, Z a *linear representation of* M.

Figure 41 illustrates Definition 50. The underlying distributed data scheme $(P_{41}, F_{41}, \equiv_\emptyset)$ is shown in Figure 41a. It contains two places A

places P_{41}:	A, B
function symbols F_{41}:	$\mathsf{f} : S \to S, \mathsf{c} :\to S$

(a) The distributed data scheme $(\mathsf{P}_{41}, \mathsf{F}_{41}, \equiv_\emptyset)$.

	A	B	
$\mathsf{a}^{41.1}$	$[\mathsf{f}(\mathbf{x})]$	$[\mathsf{f}(\mathsf{f}(\mathbf{x}))]$	
$\mathsf{z}^{41.1}$	$[\mathsf{f}(\mathsf{f}(\mathsf{c}))]$	$[\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{c})))]$	$\in \mathcal{R}\left(\mathsf{a}^{41.1}\right)$
$\mathsf{a}^{41.2}$	$[\mathsf{f}(\mathbf{x})]$	$[\mathbf{x}]$	
$\mathsf{z}^{41.2}$	$[\mathsf{f}(\mathsf{f}(\mathsf{c}))]$	$[\mathsf{f}(\mathsf{c})]$	$\in \mathcal{R}\left(\mathsf{a}^{41.2}\right)$
$\mathsf{a}^{41.3}$	$[\mathbf{x}, \mathbf{x}]$	0	
$\mathsf{s}^{41.3}$	$[\mathsf{f}(\mathsf{c}), \mathsf{f}(\mathsf{c})]$	0	$\in \mathcal{R}\left(\mathsf{a}^{41.3}\right)$
$\mathsf{a}^{41.4}$	$[\mathbf{x}, \mathbf{x}, \mathbf{x}]$	0	
$\mathsf{s}^{41.4}$	$[\mathsf{f}(\mathsf{c}), \mathsf{f}(\mathsf{c}), \mathsf{f}(\mathsf{c})]$	0	$\in \mathcal{R}\left(\mathsf{a}^{41.4}\right)$

(b) Four parameterized markings with realizations.

marking	$\in \textsc{Lin}(S, Z)$?
$\mathsf{z}^{41.1} + \mathsf{z}^{41.2}$	$\times$
$\mathsf{s}^{41.3}$	✓
$\mathsf{s}^{41.3} + \mathsf{z}^{41.1}$	✓
$\mathsf{s}^{41.3} + \mathsf{s}^{41.4} + \mathsf{z}^{41.1}$	$\times$
$\mathsf{s}^{41.4} + \mathsf{z}^{41.1} + \mathsf{z}^{41.1} + \mathsf{z}^{41.4}$	✓

(c) Linear representations from parameterized markings with $Z = \{\mathsf{a}^{41.1}, \mathsf{a}^{41.2}\}$ and $S = \{\mathsf{a}^{41.3}, \mathsf{a}^{41.4}\}$.

Figure 41.: Example of a linear representation $\textsc{Lin}(A, Z)$.

and B, and two symbols c and f over one sort S. The terms induced are c, f(c), f(f(c)), ... We will reuse this distributed data scheme in the following chapters for examples.

Figure 41b shows the four $\{\mathbf{x}\}$-parameterized markings $a^{41.1}$, $a^{41.2}$, $a^{41.3}$, and $a^{41.4}$. For each parameterized marking the realization is shown, where $\mathbf{x}$ is mapped to f(c). The realization where $\mathbf{x}$ is mapped to f(c) is shown for each parameterized marking.

In Figure 41c, we consider the finite sets of parameterized markings $S = \{a^{41.3}, a^{41.4}\}$ and $Z = \{a^{41.1}, a^{41.2}\}$ and their linear realizations $\textsc{Lin}(S, Z)$. The sum of zeros $z^{41.1} + z^{41.2}$ does not yield a linear realization, as no realization from S is contained in the sum. If we consider the single realization $s^{41.3}$, we obtain an element of $\textsc{Lin}(S, Z)$, as defined in Definition 50. Here, the empty sum is also a sum of realizations from Z. If the three realizations $s^{41.3} + s^{41.4} + z^{41.1}$ are added, we do not obtain a marking in $\textsc{Lin}(S, Z)$, as more than one realization of S is added. The last row shows the sum $s^{41.4} + z^{41.1} + z^{41.1} + z^{41.4}$, where the zero $z^{41.1}$ is added twice. The result is again a linear realization of S, Z.

6.2 COMPUTATIONAL TASKS

We break the computation of non-preserving steps up into four computational tasks. Each computational task is described and studied in a separate chapter. In Figure 42 the four computational tasks are shown in boxes with their dependency of computation as arrows. In Figure 42, we abstract from technical details, which will be studied in each respective chapter. In the following subsections we summarize each computational task.

OVERALL INPUT: t, Φ, $(P, F, \equiv)$. The overall input for the computation of non-preserving steps is an algebraic equation or algebraic inequality Φ and a transition t, both subject of the same underlying compact distributed data scheme $(P, F, \equiv)$. Hence, the distributed data scheme is given by the finite set of function symbols F and finite set of places P. The congruence $\equiv$ on terms is infinite, we assume it to be given by two computing procedures, which exist as $(P, F, \equiv)$ is compact. The first computing procedure returns a finite representation for each unification problem as input. The second computing procedure decides for each dis-unification problem if it is dis-unifiable. The algebraic equation or algebraic inequality Φ specifies data integrity and the set of satisfying markings. In general an algebraic contains a

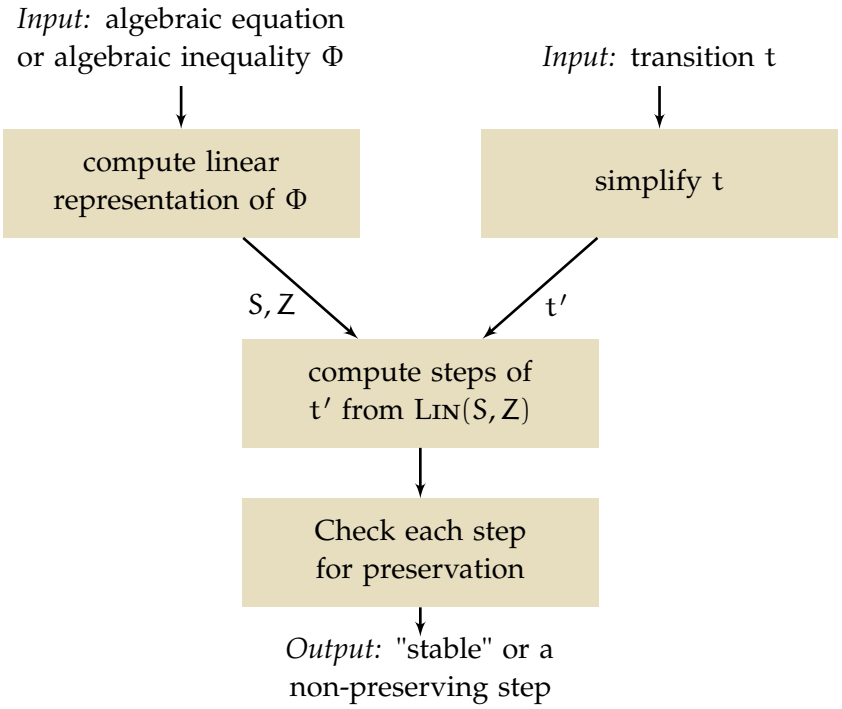


Figure 42.: The computational tasks for non-preserving step.

finite set T of transitions. In that case, we repeat the computation for each transition $t \in T$.

OVERALL OUTPUT. If there does not exist a non-preserving steps, then the computation outputs "stable". In that case, Φ is preserved by all steps induced by the transition t. Otherwise, a non-preserving step is the output. Here, the algorithm computes one non-preserving step as witness. The set of all non-preserving steps may be infinite.

6.2.1 Compute a Linear Representation

The first computational task is the computation of a linear representation of a given algebraic equation or inequality. The detailed discussion and full proofs can be found in Chapter 9. The definition of a linear representation is given in the previous section.

The core idea is that each *solution* of an algebraic equation or algebraic inequality is a sum of an *irreducible solution* and *irreducible zeros*. We study those definitions in Chapter 8 for *Diophantine* equations and inequalities and extend them in Chapter 9.

INPUT: Φ. The input for the computation is only the algebraic equation or inequality Φ. The linear representation does not depend on a transition.

OUTPUT S, Z. The output are two sets S and Z of parameterized markings where the two form a linear representation of the set of solutions: $\textsc{Lin}(S, Z) = \mathbb{L}(\Phi)$.

6.2.2 Simplify the Input Transition

The second computational task is simplification of a given transition. This computational task is studied in detail in Chapter 7. A transition is *simple*, if all its arc inscriptions are either null or monomial. The output of this computational task is a simple transition t' which induces an equivalent set of non-preserving steps. To achieve this, the distributed data scheme is adapted by introducing new places. Thus, technically the algebraic equation or algebraic inequality is also adapted. However, as this adaption is technically simple, we abstract from this aspect in this overview.

INPUT: t. The input is a transition t.

OUTPUT. The output is a transition t', which is simple.

6.2.3 Compute Steps from a Linear Representation

Given a linear representation S, Z of a set of markings, a transition may induce a set of steps starting from a realization of S, Z. In this computational task a finite representation of all steps is computed. Here, we may consider a relevant subset of the steps to compute a non-preserving step. More specifically, the output is a finite set of parameterized steps. The approach is discussed in detail in Chapter 10.

INPUT: S, Z, t. The input are sets of parameterized markings S, Z, such that $\textsc{Lin}(S, Z)$ is linear representation of the set of solutions. The other input is a simple transition t.

OUTPUT: $(a^1, b^1), \ldots, (a^n, b^n)$. The output is a finite set of parameterized steps.

6.2.4 Check Each Step for Preservation

The last computational task for the computation of non-preserving steps is the preservation check. Here, the input is a finite set of parameterized steps. For each parameterized step task is to check, whether there exists a realization such that the target marking is not satisfying. This computational task is studied in more detail in Section 10.3 of Chapter 10. The last computational task is technically simple and thus described in Section 10.3 and not in a chapter. We emphasize this task in the overview as it is illustrates the approach when considered as a separate step.

INPUT: $(a^1, b^1), \ldots, (a^n, b^n)$. The input for this computational task is a finite set of parameterized steps.

OUTPUT. The output is either a non-preserving step or "stable", if the set of non-preserving steps is empty.

6.3 DISCUSSION

We have published a proof for a restricted version of Theorem 103 in [TS16b]. The technical report with full proofs can be found in [TS16c]. For the publications [TS16b; TS16c], the author of this thesis contributed the idea for the proof. Carving out the main lemmas and organizing the proof was done in cooperation with the second author. The introduction and conclusion was written by the second author. In [TS16b; TS16c], we considered only homogeneous algebraic equations. Inequalities were not studied. Furthermore, the result was restricted to the Herbrand structure of a single sort. Theorem 103 is more general because compact data schemes are considered, and algebraic inequalities are also considered with arbitrary right-hand side.

Related Work

We will discuss related work of each computational step in each chapter separately. In this chapter, we discuss the definition of a linear representation with respect to related work.

SEMI-LINEAR SETS. Definition 50, for *linear representation* is inspired by [GS66]. Here, vectors of natural numbers are considered. In [GS66] a *semi-linear set* is defined as a finite union of *linear sets*. A linear set is a *constant vector* plus a finite sum of vectors from a finite set of *period*

vectors. However, terms and data objects are not considered in their work; the authors restrict themselves to vectors of natural numbers. The main result of [GS66] is that a set of vectors of natural numbers is semi-linear if and only if it is the solution set of a *Presburger formula*. A Presburger formula is a first order formula over natural numbers including addition.

In our setting, *Diophantine equations and inequalities* as studied in Chapter 8 build the basis of the representation of the solution space. Thus, a Diophantine equation or inequality can be seen as a special case of a Presburger formula. On the other hand, considering sums of a finite set of constant vectors and one finite set of period vectors can be seen as a special case of a semi-linear set. Accordingly, our results do not contradict with these results. Moreover, our result is not an immediate consequence.

7 SIMPLIFYING A TRANSITION

In this chapter, we show that we may assume without loss of generality, that each transition is *simple* for the computation of a non-preserving step. To achieve this, we show that a transition t can be encoded into a simple transition t' and an algebraic equation or inequality Φ can be encoded into an algebraic equation or inequality Φ' such that the following holds: Φ is stable with respect to t if and only if Φ' is stable with respect to t'. Furthermore, given a non-preserving step of t' we can compute a non-preserving step of t. This way, we can reduce the computation of a non-preserving step of Φ with respect to t into the computation of a non-preserving step of Φ' with respect to the simple transition t'. As a consequence, in the following chapters we assume each transition to be simple, which simplifies the proofs and increases their readability. The proofs shown in this chapter exploit some basic properties of transitions and polynomials such as the additivity of abstraction queries as shown in Lemma 28 (Page 68).

ACKNOWLEDGEMENT. The idea for the proof shown in this chapter was written after a discussion with Karsten Wolf who sketched the main idea.

7.1 SIMPLE TRANSITIONS

A polynomial is *simple* if the support is singleton. Accordingly, a marking or a transitions is simple if it is simple for every place.

Definition 51 (Simple)
Let $r \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F}\rangle$. Let $a \in \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^{P}$. Let $t \in \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^{P} \times \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^{P}$ be a transition.

- r is *simple* if $|\text{support}(r)| \leqslant 1$.
- a is *simple* if a_p is simple for all $p \in P$.
- t is *simple* if t^- and t^+ are simple.

For non-simple objects we can quantify the "distance" to being simple. Intuitively, the number of terms that are "too much" to be simple are counted.

Definition 52 (Unsimple Terms)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $r \in \mathbb{Z}\langle \mathbb{V}_F \rangle$. Let $a \in \mathbb{N}\langle \mathbb{V}_F \rangle^P$ be a parameterized marking. Let $t \in \mathbb{N}\langle \mathbb{V}_F \rangle^P \times \mathbb{N}\langle \mathbb{V}_F \rangle^P$ be a transition.

Then, we define:

$$\#(r) = \begin{cases} 0 & \text{, if } |\text{support}(r)| \leqslant 1 \\ |\text{support}(k)| - 1 & \text{, otherwise.} \end{cases}$$

$$\#(a) = \sum_{p \in P} \#(a_p)$$

$$\#(t) = \#(t^-) + \#(t^+)$$

We observe that a polynomial/marking/transition x is simple if and only if $\#(x) = 0$.

In the following, we encode a non-simple transition into a simple transition. To this end, we introduce new places and split the non-simple polynomials. The following definition introduces one new place to split a non-simple transition.

Definition 53 (Simplification)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $a \in \mathbb{N}\langle \mathbb{V}_F \rangle^P$ with $\#(a) \geqslant 1$. Let $p \in P$, $\theta \in \langle \mathbb{V}_F \rangle$ with:

1. $|\text{support}(a_p)| \geqslant 2$
2. $\theta \in \text{support}(a_p)$

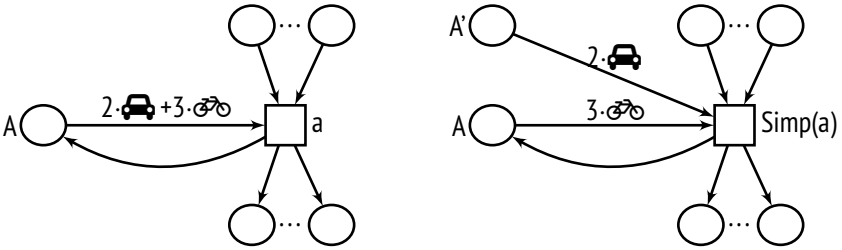
Let $\hat{p} \in \mathbb{V} \setminus P$ be a fresh location with $\textsc{Sort}(\hat{p}) = \textsc{Sort}(\theta)$. Let $\textsc{Simp}(a) \in \mathbb{N}\langle \mathbb{V}_F \rangle^{P \cup \{\hat{p}\}}$. Let for all $p' \in P \cup \{\hat{p}\}$:

$$\textsc{Simp}(a)_{p'} = \begin{cases} a_p - a_p(\theta) \cdot \theta & \text{, if } p' = p \\ a_p(\theta) \cdot \theta & \text{, if } p' = \hat{p} \\ a_{p'}(\theta) \cdot \theta & \text{, otherwise.} \end{cases}$$

Then, $\textsc{Simp}(a)$ is a *simplification of* a *over* p *and* θ *in* $\hat{p}$.

For a transition $t \in \mathbb{N}\langle \mathbb{V}_F \rangle^P \times \mathbb{N}\langle \mathbb{V}_F \rangle^P$ with $\#(t) \geqslant 1$, we extend this notion accordingly as $\textsc{Simp}(t)$ for $\big(\textsc{Simp}(t^-), t^+\big)$, or $\big(t^-, \textsc{Simp}(t^+)\big)$, respectively.

An illustration of Definition 53 is shown in Figure 43. In Figure 43a, a_A is not simple, as $\text{support}(a_A) = \{🚲, 🚗\}$ and hence $\#(a_A) = 1$. Accord-



(a) An unsimple transition. **(b)** A simpler transition.

Figure 43.: Reducing the number of terms on arcs by introducing new locations.

ingly, a is not simple and we can apply Definition 53. The result is shown in Figure 43. Regarding the manipulated arcs, we observe that Simp(a) is simple with respect to both manipulated arc inscriptions, as $|\text{support}(2 \cdot 🚗)| = 1$ and $|\text{support}(3 \cdot 🚲)| = 1$.

We observe that a simplification always exist if a marking/transition is not simple. The following lemma shows that a simplification is always simpler.

Lemma 54 (Simpler Transition)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $a \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ with $\#(a) \geqslant 1$. Let $\textsc{Simp}(a)$ be a simplification of a.
Then, $\#(\textsc{Simp}(a)) < \#(a)$.

Proof of Lemma 54. Let $\textsc{Simp}(a)$ be a simplification of a over $\hat{p}$. Then, the following statements hold for all $p' \in P \setminus \{p\}$:

$$\begin{aligned} \#(\textsc{Simp}(a)_{\hat{p}}) &= 0 \\ \#(\textsc{Simp}(a)_{p}) &= \#(a_p) - 1 \\ \#(\textsc{Simp}(a)_{p'}) &= \#(a_{p'}) \end{aligned}$$

Let $P' = P \cup \{\hat{p}\}$. Then, we have:

$$\begin{aligned} \#(\textsc{Simp}(a)) &= \sum_{p' \in P'} \#(\textsc{Simp}(a)_{p'}) \\ &= \sum_{p' \in P} \#(\textsc{Simp}(a)_{p'}) + 0 \\ &= \sum_{p' \in P} \#(a_{p'}) - 1 \\ &= \#(a) - 1 < \#(a) \end{aligned}$$

In Definition 53, for an unsimple transition t a new place is introduced for the encoding in the simpler transition $\textsc{Simp}(t)$. The distributed data schemes of $\textsc{Simp}(t)$ and t are different. We can *extend* an abstraction query over the distributed data scheme of t to an abstraction query over the distributed data scheme of $\textsc{Simp}(t)$ as follows: We keep the term-induced function and the integer coefficient for every "old" place. We copy the term-induced function and the integer coefficient for the new introduced place $\hat{p}$ from the original place p.

Definition 55 ($\hat{p}$-Extended Abstraction Query)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $\mathbf{k} = (\ell, \mathbf{o})$ be an abstraction query. Let $\underline{p} \in P$ and $\hat{p} \in \mathbb{V} \setminus P$ with $\textsc{Sort}(\hat{p}) = \textsc{Sort}(\underline{p})$. Let $\mathbf{k}' = (\ell', \mathbf{o}')$ be the abstraction query over $(P \cup \{\hat{p}\}, F, \equiv)$ such that for all $p \in P \cup \{\hat{p}\}$:

$$\mathbf{o}'_p = \begin{cases} \mathbf{o}_p & \text{, if } p \in P \\ \mathbf{o}_{\underline{p}} & \text{, if } p = \hat{p} \end{cases}$$

$$\ell'_p = \begin{cases} \ell_p & \text{, if } p \in P \\ \ell_{\underline{p}} & \text{, if } p = \hat{p} \end{cases}$$

Then, $\mathbf{k}'$ is the $\hat{p}$-extended abstraction query of $\mathbf{k}$.

In Lemma 56, we state an important property of extended abstraction queries: For a parameterized effect a and its simplification $\textsc{Simp}(a)$ over $\hat{p}$, the $\hat{p}$-extended abstraction query applied $\textsc{Simp}(a)$ gives the same result as the original abstraction on applied a, even if a substitution is applied to a.

Lemma 56 (Extended Abstraction Query and Simplification)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $p \in P$, $\theta \in \langle {}^{\mathbb{V}}_{F} \rangle$. Let $a \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle^{P}$ with $\#(a) \geqslant 1$ and $\textsc{Simp}(a)$ a simplification with p and θ over $\hat{p}$. Let $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \mathbb{V}$. Let $\mathbf{k}$ be an abstraction query and $\mathbf{k}'$ be the $\hat{p}$-extended abstraction query of $\mathbf{k}$.

Then, the following holds:

$$\mathbf{k}' \odot \sigma(\textsc{Simp}(a)) = \mathbf{k} \odot \sigma(a)$$

Proof of Lemma 56. Applying Definition 53, we have:

$$\mathbf{k}' \odot \textsc{Simp}(a) = \sum_{p' \in P \cup \{\hat{p}\}} \sigma(\ell'_{p'}, \mathbf{o}'_{p'}(\textsc{Simp}(a)_{p'}))$$

$$
\begin{aligned}
&= \sum_{p' \in P} \sigma(\ell'_{p'} \mathbf{o}'_{p'}(\textsc{Simp}(a)_{p'})) + \sigma(\ell'_{\hat{p}} \mathbf{o}'_{\hat{p}}(\textsc{Simp}(a_{\hat{p}})) \\
&= \sum_{p' \in P} \sigma(\ell'_{p'} \mathbf{o}'_{p'}(\textsc{Simp}(a)_{p'})) + \sigma(\ell_p \mathbf{o}_p(\textsc{Simp}(a_{\hat{p}}))) \\
&= \sum_{p' \in P} \sigma(\ell_{p'} \mathbf{o}_{p'}(a_{p'})) \\
&\quad - \sigma(\ell_p \mathbf{o}_p(a_p(\theta) \cdot \theta) + \sigma(\ell_p \mathbf{o}_p(a_p(\theta) \cdot \theta)) \\
&= \sum_{p' \in P} \sigma(\ell_{p'} \mathbf{o}_{p'}(a_{p'})) \\
&= \mathbf{k} \odot \sigma(a)
\end{aligned}
$$

7.2 SIMPLIFICATION AND NON-PRESERVING STEPS

In this section we show the main result of this chapter: Simplification of a transition while extending the abstraction query does not alter emptiness of the set of non-preserving steps.

To this end, we show in Lemma 57 that for every step s of a transition, there exists a step s' of the simplified transition, such that the extended abstraction query applied to s' has the same result for source and target marking of s. Furthermore, the inverse direction holds as well.

Lemma 57 (Extended Abstraction Queries Applied to Steps of Simplified Transitions)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $\underline{p} \in P$, $\hat{p} \in \mathbb{V} \setminus P$ with $\textsc{Sort}(\underline{p}) = \textsc{Sort}(\hat{p})$. Let $\theta \in \langle {}^{\mathbb{V}}_{F} \rangle$. Let $t \in \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P \times \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P$ be a transition with $\#(t) \geqslant 1$ and $\textsc{Simp}(t)$ a simplification of t over $\underline{p}$ and θ in $\hat{p}$. Let $\mathbf{k}$ be an abstraction query and $\mathbf{k}'$ the $\hat{p}$-extended abstraction query $\mathbf{k}'$. Let $r, r' \in \mathbb{Z}\langle {}^{\emptyset}_{F} \rangle$.

Then, the following statements are equivalent:

1. There exists a step $(m, m') \in \mathcal{R}(t)^{\uparrow}$ with $\mathbf{k} \odot m = r$ and $\mathbf{k} \odot m' = r'$
2. There exists a step. $(m, m') \in \mathcal{R}\left(\textsc{Simp}(t)\right)^{\uparrow}$ with $\mathbf{k}' \odot m = r$ and $\mathbf{k}' \odot m' = r'$.

Proof of Lemma 57. 1.⇒2.: There exists $\underline{m} \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$, $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ such that:

$$(m, m') = (\underline{m} + \sigma(t^-), \underline{m} + \sigma(t^+))$$

We extend $\underline{m}$ to $\hat{\underline{m}} \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P \cup \{\hat{p}\}}$ for $p \in P \cup \{\hat{p}\}$ by:

$$\hat{\underline{m}}_p = \begin{cases} \underline{m}_p & \text{, if } p \in P \\ 0 & \text{, if } p = \hat{p} \end{cases}$$

We observe that $\mathbf{k}' \odot \hat{\underline{m}} = \mathbf{k} \odot \underline{m}$. Using Lemmas 28 and 56, we deduce:

$$\begin{aligned} \mathbf{k} \odot m &= \mathbf{k} \odot (\underline{m} + \sigma(t^-)) \\ &= \mathbf{k} \odot \underline{m} + \mathbf{k} \odot \sigma(t^-) \\ &= \mathbf{k}' \odot \hat{\underline{m}} + \mathbf{k}' \odot \sigma(\textsc{Simp}(t^-)) \\ &= \mathbf{k}' \odot \left(\hat{\underline{m}} + \sigma(\textsc{Simp}(t^-))\right) \end{aligned}$$

And accordingly, we have:

$$\begin{aligned} \mathbf{k} \odot m' &= \mathbf{k} \odot (\underline{m} + \sigma(t^+)) \\ &= \mathbf{k} \odot \underline{m} + \mathbf{k} \odot \sigma(t^+) \\ &= \mathbf{k}' \odot \hat{\underline{m}} + \mathbf{k}' \odot \sigma(\textsc{Simp}(t^+)) \\ &= \mathbf{k}' \odot \left(\hat{\underline{m}} + \sigma(\textsc{Simp}(t^+))\right) \end{aligned}$$

Moreover, it holds that:

$$\left(\sigma(\textsc{Simp}(t^-)) + \hat{\underline{m}}, \sigma(\textsc{Simp}(t^+)) + \hat{\underline{m}}\right) \in \mathcal{R}\left(\textsc{Simp}(t)\right)^{\uparrow}$$

2.⇒1.: There exists $\hat{\underline{m}} \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P \cup \{\hat{p}\}}$, $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ such that:

$$\left(m, m'\right) = \left(\hat{\underline{m}} + \sigma(\textsc{Simp}(t^-)), \hat{\underline{m}} + \sigma(\textsc{Simp}(t^+))\right)$$

Then, we define $\underline{m} \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^{P}$ for $p \in P$ by:

$$\underline{m}_p = \begin{cases} \hat{\underline{m}}_p & \text{, if } p \in P \setminus \{\underline{p}\} \\ \hat{\underline{m}}_{\underline{p}} + \hat{\underline{m}}_{\hat{p}} + & \text{, if } p = \underline{p} \end{cases}$$

We observe that $\mathbf{k} \odot \underline{m} = \mathbf{k}' \odot \hat{\underline{m}}$.

Using Lemmas 28 and 56, we deduce:

$$\begin{aligned} \mathbf{k}' \odot m &= \mathbf{k}' \odot \left(\hat{\underline{m}} + \sigma(\textsc{Simp}(t^-))\right) \\ &= \mathbf{k}' \odot \hat{\underline{m}} + \mathbf{k}' \odot \sigma(\textsc{Simp}(t^-)) \\ &= \mathbf{k} \odot \underline{m} + \mathbf{k} \odot \sigma(t^-) \\ &= \mathbf{k} \odot (\underline{m} + \sigma(t^-)) \end{aligned}$$

And accordingly, we have:

$$\begin{aligned}\mathbf{k}' \odot m &= \mathbf{k}' \odot \left(\underline{\hat{m}} + \sigma(\textsc{Simp}(t^+))\right)\\ &= \mathbf{k}' \odot \underline{\hat{m}} + \mathbf{k}' \odot \sigma(\textsc{Simp}(t^+))\\ &= \mathbf{k} \odot \underline{m} + \mathbf{k} \odot \sigma(t^+)\\ &= \mathbf{k} \odot (\underline{m} + \sigma(t^+))\end{aligned}$$

Moreover, it holds that:

$$\left(\sigma(t^-) + \underline{\hat{m}}, \sigma(t^+) + \underline{\hat{m}}\right) \in \mathcal{R}(t)^\uparrow$$

As the proof is constructive, we observe that we can translate every step of t into a step of $\textsc{Simp}(t)$ and vice versa.

Lemma 57 shows the coincidence of emptiness of non-preserving steps with simplification of one term. However, applying the previous lemma inductively we observe:

Corollary 58 (Transition Simplification)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $t \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P \times \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ be a transition with $\#(t) \geqslant 1$. Let $\mathbf{k}$ be an abstraction query. Let $r, r' \in \mathbb{Z}\langle \frac{\emptyset}{F} \rangle$.

Then, there exist a distributed data scheme $(P', F, \equiv)$ and a simple transition t', and an abstraction query $\mathbf{k}'$ such that the following properties are equivalent:

1. There exist a step $(m, m') \in \mathcal{R}(t)^\uparrow$ with: $\mathbf{k} \odot m = r$ and $\mathbf{k} \odot m' = r'$.
2. There exist a step $(m, m') \in \mathcal{R}(t')^\uparrow$ with: $\mathbf{k}' \odot m = r$ and $\mathbf{k}' \odot m' = r'$.

Proof of Corollary 58. We may apply Lemma 57 inductively, as $\#(t)$ is finite and $\#(\textsc{Simp}(t)) < \#(t)$ by Lemma 54.

Now we can state the main result of this chapter in Theorem 59: For a given transition t and algebraic equation or inequality Φ, we can encode the problem of the existence of non-preserving steps into the existence of non-preserving steps of an according algebraic equation or inequality Φ' and simple transition t'.

Theorem 59 (Reduction to Simple Transition)
Let $(P, F, \equiv)$ be a distributed data scheme. Let $t \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P \times \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ be a transition. Let $\mathbf{k}$ be an abstraction query and $r \in$

$\mathbb{Z}\langle {}^{\emptyset}_{F}\rangle$ a polynomial. Let E be the algebraic equation $\mathbf{k} \odot P \doteq r$. Let I be the algebraic inequality $\mathbf{k} \odot P \dot{\geqq} r$. Let $\Phi \in \{E, I\}$.

Then, there exists a set of places $P' \supset P$, a simple transition $t' \in \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^{P'} \times \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^{P'}$ and algebraic equation or inequality Φ' such that the following statements are equivalent:

1. There exists a step of t that is not Φ-preserving.
2. There exists a step of t' that is not Φ'-preserving.

Proof of Theorem 59. As defined in Definition 53, we can simplify t until we receive a simple transition t' and accordingly extend $\mathbf{k}$ to $\mathbf{k}'$. Now we show the equivalence:

1.⇒2.: Let $(m, m') \in \mathcal{R}(t)^{\uparrow}$ be a step that is not Φ-preserving. Then by Corollary 58, there exists a step $(\hat{m}, \hat{m}') \in \mathcal{R}(t)^{\uparrow})$ with $\mathbf{k} \odot m = \mathbf{k}' \odot \hat{m}$ and $\mathbf{k} \odot m' = \mathbf{k}' \odot \hat{m}$. As the right-hand side of Φ and Φ' is the same, the step $(\hat{m}, \hat{m}')$ is not Φ'-preserving.

2.⇒1.: Let $(\hat{m}, \hat{m}') \in \mathcal{R}(t')^{\uparrow}$ be a step that is not Φ'-preserving. Then by Corollary 58, there exists a step $(m, m') \in \mathcal{R}(t)^{\uparrow})$ with $\mathbf{k} \odot m = \mathbf{k}' \odot \hat{m}$ and $\mathbf{k} \odot m' = \mathbf{k}' \odot \hat{m}$. Again, as the right-hand side of Φ and Φ' is the same, the step (m, m') is not Φ-preserving.

Regarding the complexity, we observe that we have to introduce a new place for every non-simple term and copy the coefficients accordingly for this transition. Hence, the size of simple problem is polynomial in the size of the original problem. Moreover, the proof shows that we can easily translate steps by summing along the new introduced places.

As a summary, in the following chapters, we assume the input transition to be simple. Theorem 59 justifies this assumption.

8 SOLUTIONS OF LINEAR DIOPHANTINE EQUATIONS AND INEQUALITIES

In this chapter, we study the solutions of *linear Diophantine equations* and *linear Diophantine inequalities*. We show each solution is a sum of an *irreducible solutions* and finitely many *irreducible zeros*. The main result is Theorem 63, which states that each irreducible solution is bounded in size. Moreover, the bound is polynomial in the size of the equation or inequality, respectively.

In Section 8.1 we recall the definitions necessary to state the main result of this section in Theorem 63. The proof of Theorem 63 is separated into two sections: In Section 8.2, we prove Theorem 63 for linear Diophantine equations. In Section 8.3, we prove Theorem 63 for linear Diophantine inequalities. We discuss related work in Section 8.4.

8.1 DIOPHANTINE EQUATIONS AND INEQUALITIES

In this section, we recall the definitions of reducible and irreducible solutions and zeros of linear Diophantine equations and inequalities. We state the main result of this chapter in Theorem 63.

Intuitively, a linear Diophantine equation or inequality is a vector of integer coefficients together with a right-hand side. With each coefficient comes a symbol for an unknown.

Definition 60 ((Linear) Diophantine Equation, (Linear) Diophantine Inequality)
Let $n \in \mathbb{N}$. Let $\kappa \in \mathbb{Z}^n$, $r \in \mathbb{Z}$. Let x_i $(1 \leqslant i \leqslant n)$ be symbols for unknowns.

Then,

- D : $\sum_{i=1}^{n} \kappa_i x_i \doteq r$ is a *Diophantine equation* and
- D′ : $\sum_{i=1}^{n} \kappa_i x_i \dot{\geqslant} r$ is a *Diophantine inequality*

with dimension $n \in \mathbb{N} \setminus \{0\}$, coefficients $\kappa \in \mathbb{Z}^n$, and right-hand side $r \in \mathbb{Z}$.

In this chapter, we only consider *linear* Diophantine equations and inequalities. Therefore, we write simply Diophantine equation or in-

$$E_1 : \qquad 11x_1 - 13x_2 = 2$$

(a) The linear Diophantine equation E_1.

x_1	x_2	solution?	reducible solution?	zero?
12	10	✓	×	×
13	11	×	-	✓
25	21	✓	✓	×

(b) An irreducible solution, a zero, and a reducible solutions.

Figure 44.: Example of Diophantine equation with solutions and a zero.

equality, implicitly referring to linear Diophantine equations or inequalities.

Figure 44a shows the Diophantine equation E_1, which has two coefficients: 11 and 13. The right-hand side is 2. Figure 45a shows the Diophantine inequality I_1, which has three coefficients: 2, 3, and -1. The right-hand side is 8.

We define solutions and zeros for a Diophantine equation or inequality.

> **Definition 61** (Solutions and Zeros of Diophantine Equations and Inequalities)
> Let d be Diophantine equation with dimension $n \in \mathbb{N} \setminus \{0\}$, coefficients $\kappa \in \mathbb{Z}^n$, and a. Let $m \in \mathbb{N}^n$. Then,
>
> - m is a *solution* for $\sum_{i=1}^{n} \kappa_i x_i \doteq r$, if $\sum_{i=1}^{n} \kappa_i m_i = r$, and
> - m is a *solution* for $\sum_{i=1}^{n} \kappa_i x_i \dot{\geqslant} r$, if $\sum_{i=1}^{n} \kappa_i m_i \geqslant r$.
>
> Let $z \in \mathbb{N}^n$. Then,
>
> - z is a *zero* for $\sum_{i=1}^{n} \kappa_i x_i \doteq r$, if $\sum_{i=1}^{n} \kappa_i z_i = 0$, and
> - z is a *zero* for $\sum_{i=1}^{n} \kappa_i x_i \dot{\geqslant} r$, if $\sum_{i=1}^{n} \kappa_i z_i \geqslant 0$.

In Figure 44b, the first row shows a solution for E_1, as $12 \cdot 11 - 13 \cdot 10 = 132 - 130 = 2$. Additionally, the second row shows a zero for E_1 as $13 \cdot 11 - 11 \cdot 13 = 143 - 143 = 0$.

In Figure 45b, the first row shows a solution for I_1, as $2 \cdot 4 = 8 \geqslant 8$. The third row shows a zero of I_1 as $1 \cdot 2 = 2 \geqslant 0$.

Definition 60 does not consider inequalities with $\leqslant$. However, we recall that introducing $\leqslant$ would not provide greater expressive power:

$$I_1: \qquad 2x_1 + 3x_2 - x_3 \geqslant 8$$

(a) The linear Diophantine inequality I_1.

x_1	x_2	x_3	solution?	reducible solution?	zero?
4	0	0	✓	×	✓
0	3	0	✓	×	✓
1	0	0	×	-	✓
1	0	1	×	-	✓
0	1	2	×	-	✓
4	1	2	✓	✓	✓

(b) Reducible solutions, an irreducible solution and zeros of I_1.

Figure 45.: Example of Diophantine inequality with solutions and zeros.

By multiplying every coefficient and the right-hand side with -1 there exist always equivalent inequalities with $\geqslant$. Therefore, we restrict ourselves to $\geqslant$ and avoid notational overhead.

A solution may be a sum of a solution and a zero. We call a solution *reducible*, if it can be decomposed in a non-trivial solution and a non-trivial zero.

> **Definition 62** (Irreducible Solution)
> Let d be a *Diophantine equation* and d' be a *Diophantine inequality*. Let $m \in \mathbb{N}^n$ be a solution for d (d').
>
> Then, m is *reducible*, if there exists $m', m'' \in \mathbb{N}^n$ and $i, j \in \{1, \ldots, n\}$
>
> - $m'_i \neq 0$
> - $m''_j \neq 0$
> - $m = m' + m''$,
> - m' is a solution for d (d'), and
> - m'' is a zero for d (d').
>
> Accordingly, m is *irreducible*, if it is not reducible.

In Figure 44b, the third row shows an example for a reducible solution of E_1: It is a sum of the first row, which shows a solution and the second row, which shows a zero.

In Figure 45b, the sixth row shows a reducible solution of I_1: Summing the numbers of the first row, which is a solution of I_1 and the fifth row, which is a zero of I_1, we obtain the sixth row. Contrary to that, the first row shows an irreducible solution of I_1, as every $x \in \mathbb{N}^3$ with $x_1 < 4$ and $x_2 = x_3 = 0$ is not a solution of I_1: E.g. if $x_1 = 3$, we have $3 \cdot 2 = 6 \not\geqslant 8$. Moreover, as $8 \geqslant 0$, we observe that every solution of I_1 is also zero of I_1. However, the converse is not true, as the third to fifth row show zeros of I_1, which are no solutions of I_1.

Now, we are able to state the main theorem of this chapter. It states that every irreducible solution is polynomial in the size of the Diophantine equation or inequality.

Theorem 63 (Polynomial Bound of Irreducible Solutions)
Let D be a Diophantine equation or inequality with dimension $n \in \mathbb{N}$, coefficients $\kappa \in \mathbb{Z}^n$, and right-hand side $r \in \mathbb{Z}$. Let $s \in \mathbb{N}^n$ be an irreducible solution of D.

Then, $\|s\| \leqslant 2n\hat{\kappa}^2 + |r|$, where $\hat{\kappa} = \max\{|\kappa_i| \mid 1 \leqslant i \leqslant n\}$.

Proof of Theorem 63. Follows from Lemma 67 (Section 8.2) for Diophantine equations and Lemma 68 (Section 8.3) for Diophantine inequalities.

8.2 BOUND FOR IRREDUCIBLE SOLUTIONS OF LINEAR DIOPHANTINE EQUATIONS

In this section, we prove Theorem 63 for equations. To this end, we now consider a fixed Diophantine equation d with dimension $n \in \mathbb{N} \setminus \{0\}$, coefficients $\kappa \in \mathbb{Z}^n$ and right-hand side $r \in \mathbb{Z}$.

First, we introduce a notation for a partition of the set of indexes based on the sign of the respective coefficient:

Definition 64 (Index Partition)
Let $n \in \mathbb{N} \setminus \{0\}$. Let $k_i \in \mathbb{Z}$ for $1 \leqslant i \leqslant n$.

Then, we define:

- $I_\kappa^+ = \{i \in \mathbb{N} \mid 1 \leqslant i \leqslant n \text{ and } \kappa_i > 0\}$
- $I_\kappa^- = \{i \in \mathbb{N} \mid 1 \leqslant i \leqslant n \text{ and } \kappa_i < 0\}$
- $I_\kappa^0 = \{i \in \mathbb{N} \mid 1 \leqslant i \leqslant n \text{ and } \kappa_i = 0\}$

We observe that $I_\kappa^+ \cup I_\kappa^- \cup I_\kappa^0 = \{1, \ldots, n\}$.

Now we state a sufficient criterion for reducibility of a solution m. The prerequisite is that there exists one strict positive coefficient κ_j with $j \in I_\kappa^+$ and one strict negative coefficient κ_ℓ with $\ell \in I_\kappa^-$ and the values of m_j,m_ℓ are higher than the coefficient with the respective other index.

Lemma 65 (Reducibility Criterion)
Let $m \in \mathbb{N}^n$ be a solution of d. Let $j \in I_\kappa^+$, $\ell \in I_\kappa^-$ with

- $m_j > |\kappa_\ell|$ and
- $m_\ell > \kappa_j$.

Then, m is reducible.

Proof of Lemma 65. We define $m' \in \mathbb{N}^n$ for $i \in \{1, \dots, n\}$:

$$m'_i = \begin{cases} |\kappa_\ell| & \text{, if } i = j \\ \kappa_j & \text{, if } i = \ell \\ 0 & \text{, otherwise.} \end{cases}$$

Then,

$$\sum_{i=1}^{n} \kappa_i m'_i = \kappa_j \kappa_\ell + |\kappa_\ell| \, \kappa_j = \kappa_j \kappa_\ell - \kappa_j \kappa_\ell = 0 \, .$$

As $m' < m$, there exists $m'' \in \mathbb{N}^n$ with $m'' = m - m'$. Then, m' is a solution:

$$\sum_{i=1}^{n} \kappa_i m''_i = \sum_{i=1}^{n} \kappa_i m_i - \sum_{i=1}^{n} \kappa_i m'_i = \sum_{i=1}^{n} \kappa_i m_i = r \, .$$

Moreover, it holds that $m = m' + m''$ and both m' and m'' are not zero for all indexes and thus m is reducible.

We illustrate Lemma 65 with Figure 46. The Diophantine equation D_{46} with two unknowns, x_1 and x_2, is shown in Figure 46a. Figure 46b shows two solutions and a zero. Here, for $m^{46.3}$ holds that $m_1^{46.3} = 5 > 4$ and $m_2^{46.3} = 4 > 3 = |-3|$. Hence, $m^{46.3}$ satisfies the prerequisite of Lemma 65 and is reducible. As described in the proof, we can identify the zero $m^{46.2}$ and solution $m^{46.1}$. It holds that $m^{46.3} = m^{46.2} + m^{46.1}$, and hence $m^{46.3}$ is reducible.

Taking the contrapositive of Lemma 65, we get a necessary condition for irreducibility in the following lemma:

$$4x_1 - 3x_2 = 1$$

(a) The Diophantine equation D_{46}.

	x_1	x_2	solution?	reducible solution?	zero?
$m^{46.1}$	1	1	✓	×	×
$m^{46.2}$	3	4	×	-	✓
$m^{46.3}$	4	5	✓	✓	×

(b) Some solutions and a zero of D_{46}.

Figure 46.: Example for reducibility criterion of Lemma 65.

Lemma 66 (Irreducibility Criterion)
Let $I_\kappa^+ \neq \emptyset$ and $I_\kappa^- \neq \emptyset$. Let $m \in \mathbb{N}^n$ be an irreducible solution. Then, one of the following statements holds:

1. $\max\{m_i \mid i \in I_\kappa^-\} \leqslant \min\{\kappa_i \mid i \in I_\kappa^+\}$
2. $\max\{m_i \mid i \in I_\kappa^+\} \leqslant \min\{|\kappa_i| \mid i \in I_\kappa^-\}$

Proof of Lemma 66. It is sufficient to show that if 1. and 2. are not satisfied, we may apply Lemma 65. To this end, let $j \in I_\kappa^+$ such that:

$$m_j > \min\{|\kappa_i| \mid i \in I_\kappa^-\}.$$

Moreover, let $\ell \in I_\kappa^-$ such that:

$$m_\ell > \min\{\kappa_i \mid i \in I_\kappa^+\}$$

By the minima, we have that $m_j > |\kappa_\ell|$ and $m_\ell > \kappa_j$ and hence, we may apply Lemma 65.

Now, we use Lemma 66 to prove the first part of Theorem 63 in the following lemma.

Lemma 67 (Bound for Irreducible Solutions of Equations)
Let $n \in \mathbb{N} \setminus \{0\}$. Let $\sum_{i=1}^n \kappa_i x_i = r$ be a linear Diophantine equation. Let $\hat{\kappa} = \max\{|\kappa_i| \mid 1 \leqslant i \leqslant n\}$. Let m be an irreducible solution.

Then, $\sum_{i=1}^n m_i \leqslant 2n\hat{\kappa}^2 - r$.

Proof of Lemma 67. First, we observe for all $i \in \{1, \dots, n\}$:

$$\kappa_i = 0 \text{ implies } m_i = 0 \qquad (*)$$

Otherwise, we could subtract the non-trivial zero $z \in \mathbb{N}^n$ with:

$$z_i = \begin{cases} m_i & \text{, if } \kappa_i = 0 \\ 0 & \text{, if } \kappa_i \neq 0 \end{cases}$$

Then, m would be reducible.

Hence, we assume w.l.o.g. that $I_\kappa^0 = \emptyset$ and hence $I_\kappa^+ \cup I_\kappa^- = \{1, \dots, n\}$.

In the remaining, we distinguish the following three cases:

1st case: $I_\kappa^+ = \emptyset$.
Now, we look at an equivalent linear Diophantine equation, defined by $\kappa'_i = -\kappa_i$ for all $i \in \{1, \dots, n\}$ and $r' = -r$. Then m is a (reducible) solution for $\sum_{i=1}^n \kappa_i x_i = r$ if and only if m is (reducible) solution for $\sum_{i=1}^n \kappa'_i x_i = r'$.

We observe that $I_{\kappa'}^- = I_\kappa^+ = \emptyset$. Hence, we reduce this case to the following case.

2nd case: $I_\kappa^- = \emptyset$.
As all coefficients are positive, we can reason with monotonicity:

$$\sum_{i=1}^n m_i \leqslant \sum_{i=1}^n \kappa_i m_i = r \leqslant |r| \leqslant \underbrace{2n\hat{\kappa}}_{\geqslant 0} + |r|$$

3rd case: $I_\kappa^+ \neq \emptyset$ and $I_\kappa^- \neq \emptyset$.
We apply Lemma 66. Hence, one of the following cases is true:

1st case: $\max\{m_i \mid i \in I_\kappa^+\} \leqslant \min\{|\kappa_i| \mid i \in I_\kappa^-\}$.
Again, we consider the equivalent linear Diophantine equation $\sum_{i=1}^n \kappa'_i x_i = r'$ with $\kappa'_i = -\kappa_i$ for all $i \in \{1, \dots, n\}$ and $r' = -r$. The set of (reducible) solutions for $\sum_{i=1}^n \kappa_i x_i = r$ and $\sum_{i=1}^n \kappa'_i x_i = r'$ are the same. Moreover, we have:

$$\begin{aligned} \max\{m_i \mid i \in I_{\kappa'}^-\} &= \max\{m_i \mid i \in I_\kappa^+\} \\ &\leqslant \min\{|\kappa_i| \mid i \in I_\kappa^-\} \end{aligned}$$

$$= \min\left\{\left|\kappa_i'\right| \mid i \in I_\kappa^+\right\}$$
$$= \min\left\{\kappa_i' \mid i \in I_\kappa^+\right\}.$$

Hence, we reduce this case to the following case.

2nd case: $\max\left\{m_i \mid i \in I_\kappa^-\right\} \leqslant \min\left\{\kappa_i \mid i \in I_\kappa^+\right\}$.
This implies:

$$m_i \leqslant \hat{\kappa} \quad \text{(for all } i \in I_\kappa^-\text{)} \tag{$**$}$$

By $*$, we have:

$$\sum_{i=1}^{n} m_i = \sum_{i \in I_\kappa^+} m_i + \sum_{i \in I_\kappa^-} m_i \tag{$***$}$$

As m is a solution, it holds that $\sum_{i=1}^{n} \kappa_i m_i = r$. Applying $***$ yields: $\sum_{i \in I_\kappa^+} \kappa_i m_i + \sum_{i \in I_\kappa^-} \kappa_i m_i = r$, which is equivalent to

$$\sum_{i \in I_\kappa^+} \kappa_i m_i = r - \sum_{i \in I_\kappa^-} \kappa_i m_i \tag{$****$}$$

$$\begin{aligned}
\sum_{i=1}^{n} m_i &\underset{***}{=} \sum_{i \in I_\kappa^+} m_i + \sum_{i \in I_\kappa^-} m_i \\
&\leqslant \sum_{i \in I_\kappa^+} \kappa_i m_i + \sum_{i \in I_\kappa^-} |\kappa_i|\, m_i \\
&\underset{****}{=} r - \sum_{i \in I_\kappa^-} \kappa_i m_i + \sum_{i \in I_\kappa^-} |\kappa_i|\, m_i \\
&= r + 2 \sum_{i \in I_\kappa^-} |\kappa_i|\, m_i \\
&\underset{**}{\leqslant} r + 2 \sum_{i \in I_\kappa^-} |\kappa_i|\, \hat{\kappa} \\
&\leqslant r + 2 \sum_{i \in I_\kappa^-} \hat{\kappa}^2 \\
&\leqslant |r| + 2 \sum_{i=1}^{n} \hat{\kappa}^2 \\
&\leqslant |r| + 2n\hat{\kappa}^2
\end{aligned}$$

8.3 BOUND FOR IRREDUCIBLE SOLUTIONS OF LINEAR DIOPHANTINE INEQUALITIES

In this section, we prove Theorem 63 for inequalities. To this end, we now consider a fixed Diophantine inequality d with dimension $n \in \mathbb{N} \setminus \{0\}$, coefficients $\kappa \in \mathbb{Z}^n$ and right-hand side $r \in \mathbb{Z}$. Moreover, we use the notation introduced in Definition 64 on Page 132: I_κ^+ and I_κ^-.

The proof of this lemma applies Lemma 67 from Section 8.2.

Lemma 68 (Bound for Irreducible Solutions of an Inequality)
Let $n \in \mathbb{N} \setminus \{0\}$. Let $\sum_{i=1}^{n} \kappa_i x_i \geqslant r$ be a linear Diophantine inequality. Let $\hat{\kappa} = \max\{|\kappa_i| \mid 1 \leqslant i \leqslant n\}$. Let m be an irreducible solution.

Then, $\sum_{i=1}^{n} m_i \leqslant 2n\hat{\kappa}^2 + |r|$.

Proof of Lemma 68. This is an indirect proof. Hence, let m be a solution with $\sum_{i=1}^{n} m_i \geqslant 2n\hat{\kappa}^2 + |r|$. We show that m is reducible. Let $s = \sum_{i=1}^{n} \kappa_i m_i$. We distinguish two cases:

1st case: $|r| \geqslant |s|$.
Then, we consider the Diophantine equation d':

$$\sum_{i=1}^{n} \kappa_i x_i = s$$

with unknowns x_i ($1 \leqslant i \leqslant n$). Then, m is a solution for the equation d'. Moreover, it holds that:

$$\sum_{i=1}^{n} m_i > 2n\hat{\kappa}^2 + |r| \geqslant 2n\hat{\kappa}^2 + |s|$$

Applying the contrapositive of Lemma 67, we deduce that m is a reducible solution of d'. Hence, there exists $m', m'' \in \mathbb{N}^n$ with:

- $m = m' + m''$,
- $\sum_{i=1}^{n} \kappa_i m_i' = 0$ (i.e. m' is a zero of d'), and
- $\sum_{i=1}^{n} \kappa_i m_i'' = s$ (i.e. m'' is a solution of d').

As $s \geqslant r$, we deduce that m'' is also a solution for the inequality d. Moreover m' is not only a zero of d', but also a zero of d. Hence, m is also a reducible solution of d.

2^{nd} case: $|r| < |s|$.
As $s \geqslant r$, we deduce that $s > 0$. Hence, there exists a $j \in I_\kappa^+$ with $m_j \geqslant 1$. We now define $m' \in \mathbb{N}^n$ for $i \in \{1, \dots, n\}$ by:

$$m'_i = \begin{cases} 1 & \text{, if } i = j \\ 0 & \text{, if } i \neq j \end{cases}$$

1^{st} case: $s - \kappa_j \geqslant r$.
We observe that $m_i \geqslant m'_i$ for all $i \in \{1, \dots, n\}$. Hence, there exists $m'' \in \mathbb{N}^n$ with $m = m' + m''$ and it holds that m'' is a solution:

$$\sum_{i=1}^{n} \kappa_i m''_i = \sum_{i=1}^{n} \kappa_i m_i - \kappa_j = s - \kappa_j \geqslant r\,.$$

Moreover, m' is a zero, as $\kappa_j \geqslant 0$. Hence m is reducible.

2^{nd} case: $s - \kappa_j < r$.
Let $s' = s - \kappa_j$. We consider the linear Diophantine equation $e := \sum_{i=1}^{n} \kappa_i x_i = s'$. We observe that $m'' = m - m'$ is a solution for e. Moreover, we have $\sum_{i=1}^{n} m''_i = \sum_{i=1}^{n} m_i - 1 > 2n\hat{\kappa}^2 + |r| - 1 > 2n\hat{\kappa}^2 + |s| - \kappa_j - 1 \geqslant 2n\hat{\kappa}^2 + |s| - \kappa_j \geqslant 2n\hat{\kappa}^2 + |s - \kappa_j|$. Again, we apply the contrapositive of Lemma 67 and see that m' is a reducible solution of e. Hence, there exists a $m'' \in \mathbb{N}^n \setminus \{0\}$ with $m'' < m' < m$ and $\sum_{i=1}^{n} \kappa_i m_i = 0$. Hence, m'' is also a zero for the Diophantine inequality d. And there exists $m''' \in \mathbb{N}^n$ with $m = m'' + m'''$ and it holds that:

$$\begin{aligned} \sum_{i=1}^{n} \kappa_i m'''_i &= \sum_{i=1}^{n} \kappa_i m_i - \sum_{i=1}^{n} \kappa_i m'_i \\ &= \sum_{i=1}^{n} \kappa_i m_i - 0 \\ &= s \\ &\geqslant r\,. \end{aligned}$$

Hence, m''' is a solution of d, m'' is a zero of d, and m is a reducible solution of d.

8.4 DISCUSSION

The main result of this chapter is Theorem 63: Each irreducible solution and each irreducible zero is bounded by a polynomial in the size of the Diophantine equation or inequality. This result is the basis for the computation of a linear representation of the set of solutions of an algebraic equation or inequality. We will use Theorem 63 in the following chapter to prove Lemmas 89 and 90 and Theorem 91.

The results are strongly related to existing work. In the following, we discuss related work in the following fields:

1. integer linear programming
2. Presburger arithmetic
3. additive number theory
4. Frobenius problem
5. lattice points in polyhedra

INTEGER LINEAR PROGRAMMING. *Linear programming* is optimization of a linear function in a real vector space subject to linear constraints. Integer linear programming is linear programming restricted to integer solutions.

According to [Dan63], the research and methods in linear programming started in 1947. In 1958, Ralph E. Gomory, improved the techniques for natural number drawing the line to solutions of linear Diophantine inequalities [Gom02]. The rise of electronic computers established (integer) linear programming to a standard technique in computer science and programming [Sch99; Jün+10; Mel17; And15].

In [Pap81], the complexity of integer linear programming and several variants is studied. The main result, Theorem 1, is proven using Cramer's rule: Every integer linear program that has a solution, has a solution which is bounded in size (in a polynomial). This result is very similar to the result obtained in this chapter. However, it is not discussed that larger solutions are the sum of irreducible solutions and irreducible zeros and that all irreducible solutions are bounded by a quadratic polynomial. Although the contributions in [Pap81] are slightly different, it is open question whether the proofs could be adapted to show Theorem 63.

PRESBURGER FORMULAS. Presburger formulas are named by Mojżesz Presburger who introduced the idea in 1929 [Sta84]. Presburger formulas are first-order formulas over natural numbers with addition. A Diophantine equation D can be seen as a special Presburger

formula ϕ, where all unknowns of D are free variables. Then, the solutions sets are equivalent. By [GS66], then the solution set is D semi-linear. However, the results on Presburger formulas are not applicable to prove the bound of the size as in Theorem 63. Moreover, it has been shown that the time complexity of the problem of satisfiability is doubly exponential [FR74].

THE FROBENIUS PROBLEM. The *Frobenius problem*, which is also known as *coin problem*, can be stated as follows: Given relatively prime natural numbers $a_1, \ldots, a_n$, find the largest natural number that is not that is not representable as a non-negative integer combination of $a_1, \ldots, a_n$ [Alf09]. As solutions are vectors of natural numbers, the problem is strongly related to find a bound for the size of irreducible solutions. However, we consider also integer coefficients and coefficients which are not relatively prime. Moreover, in the chapter, we have proven a bound for the size of solutions. In the research field of the Frobenius problem, the question is primarily for procedures and complexities results to obtain the Frobenius number.

ADDITIVE NUMBER THEORY. Lagrange stated that every nonnegative integer is the sum of four squares. Spoken differently: The set of squares is an *additive base of order 4*. Additive number theory now studies from which sets of integers it is possible to construct other integers by addition [Nat96b; Nat96a]. However, the results are not applicable as additive number theory asks for infinite sets, which members are added for bounded number of times (4 in the case of Lagrange). The most famous problem is the Goldbach conjecture: Whether the set of prime numbers are a base. However, in our setting we consider a finite number of integers: The coefficients of the equation or inequality, but a addition maybe unbounded. Hence, the results or concepts are not applicable to these results.

LATTICE POINTS IN POLYHEDRA. A Diophantine equation (inequality) can be relaxed to rational solutions. Then, the solution set is polyhedron and the intersection with the lattice of integers results in the solution set. Considering only non-negative solutions does not affect this, as the intersection is still a polyhedron. Hence, this approach can be seen as an alternative approach to integer programming [AWW02]. In [Coo+92; BHL92], the authors proved bounds on the number of lattice points. However, a bound on the number does not directly infer a semi-linear representation. Moreover, it is not obvious how to infer a bound on the size of the points from the number of points.

9 LINEAR REPRESENTATION OF SOLUTIONS

In this section, we characterize the set of solutions of an algebraic equation or algebraic inequality. We show that a linear representation of the set of solutions is computable, if the underlying data scheme is compact. The definition of a linear representation and its role for the computation of non-preserving steps has been stated in Chapter 6.

For this section, we fix the following notations:

- a distributed data scheme $(P, F, \equiv)$ (Definition 8, Page 41)
- an abstraction query $\mathbf{k} = (\boldsymbol{\ell}, \mathbf{o})$ (Definition 27, Page 66)
- a term polynomial $r \in \mathbb{Z}\langle {}^{\emptyset}_{F} \rangle$
- the algebraic equation E: $\mathbf{k} \odot P \doteq r$ (Definition 29, Page 69)
- the algebraic inequality I: $\mathbf{k} \odot P \dot{\geqq} r$ (Definition 29, Page 69)
- $\Phi \in \{E, I\}$

In this chapter, we switch the notation for polynomials. In the last chapter we used the "set-like" notation in brackets. To improve the readability of the proofs, we write polynomials as sums of monomials. As an example, let θ and ξ be terms. Then, $2 \cdot \theta + 1 \cdot \xi$ and $[\theta, \theta, \xi]$ refer to the same polynomial.

9.1 IRREDUCIBLE SOLUTIONS AND IRREDUCIBLE ZEROS

In the previous chapter, we defined irreducible solutions and irreducible zeros of a linear *Diophantine* equation or inequality. In this section, we define irreducible solutions and irreducible zeros of an *algebraic* equation or inequality.

If the right-hand side r is zero, the equation or inequality is *homogenous*. Each algebraic equation and inequality has a homogeneous variant.

Definition 69 (Homogeneous)
If $r = 0$, then E (I) is *homogeneous*.
Let H_E be the algebraic equation $\mathbf{k} \odot P \doteq 0$. Let H_I be the algebraic inequality $\mathbf{k} \odot P \dot{\geqq} 0$.
Then, H_E (H_I) is the *homogeneous variant* of E (I).

Based on the Definition 29 (Page 69), we define zeros as solutions of the homogeneous variant.

Definition 70 (Zero)
Let $z \in \mathbb{L}(H_\Phi)$ be a solution of the homogeneous variant of Φ. Then, z is a *zero* of Φ.
We denote the *set of all zeros of* Φ by $\mathbb{L}_0(\Phi)$.

If a solution is a sum of a non-trivial zero and a non-trivial solution, it is *reducible*.

Definition 71 ((Ir-)Reducible Solution)
Let $a \in \mathbb{L}(\Phi)$ be a solution of Φ. If there exist $a' \in \mathbb{L}(\Phi) \setminus \{0\}$ and $z \in \mathbb{L}_0(\Phi) \setminus \{0\}$ with $a = a' + z$, then a is *reducible*.
Otherwise, a is *irreducible*.
We denote the *set of all irreducible solutions* by $\mathbb{L}^{IR}(\Phi)$.

Combining Definition 70 and 71, we observe: *(Ir)reducible zeros* of Φ are (ir)reducible solutions of the homogeneous variant of Φ. Accordingly, we denote the *set of all irreducible zeros* by $\mathbb{L}_0^{IR}(\Phi)$.

The following lemma states that every solution is a sum of an irreducible solution and irreducible zeros.

Lemma 72 (Reducibility of Solutions)
Let $a \in \mathbb{L}(\Phi)$.
Then there exists $n \in \mathbb{N}$, $a', z^1, \dots z^n \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ such that:

1. $a = a' + \sum_{i=1}^{n} z^i$,
2. $a' \in \mathbb{L}^{IR}(\Phi)$, and
3. $z^1, \dots, z^n \in \mathbb{L}_0^{IR}(\Phi)$.

Proof of Lemma 72. As $\leqslant$ is a well-founded order on $\mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$, we apply Noetherian induction in this proof. Let $a = 0$. If a is a solution, then it is an irreducible solution, because there does not exists a zero with $0 < z < 0 = a$. Then, we choose $a' = a = 0$, $n = 0$, and the three assertions follow immediately.

For the rest of the proof, we assume $a > 0$. We distinguish the cases whether a is reducible or irreducible.

1st case: a' is irreducible.
We choose $n = 0$ and $a' = a$. The three assertions follow immediately.

2nd case: a' is reducible.
Then, by Definition 71 there exists $a'' \in \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P$ and $z \in \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P$ such that:

a) $a' \in \mathbb{L}(\Phi)$,

b) $z \in \mathbb{L}_0(\Phi) \setminus \{0\}$, and

c) $a = a'' + z$

As $0 < z < a$, it follows that $a' < a$. Hence, we may apply induction and assume a' is a sum of a irreducible solution and a finite set of irreducible zeros. It remains to show that z is a sum of a finite set of irreducible zeros. To this end, we recall that z is a solution of the homogeneous variant of Φ. Hence, we may again apply induction using z as a solution of the homogeneous variant H_Φ and $z < a$.

In Figure 47 we illustrate Definition 69 to 71 and Lemma 72. Figure 47a shows the distributed data scheme $(P_{47}, F_{47}, \equiv_\emptyset)$, which we use for all examples in this chapter. It contains two places A and B, and two symbols c and f over one sort S. The terms induced are c, f(c), f(f(c)), ...

In Figure 47b the algebraic equation E_{47} over the distributed data scheme $(P_{47}, F_{47}, \equiv_\emptyset)$ is shown. Figure 47c shows some solutions and zeros of E_{47}. $m^{47.1}$ equals 0 and thus is a zero. As no marking $m < m^{47.1}$ exists, $m^{47.1}$ is an irreducible zero. $m^{47.2}$ is a solution as $2 \cdot 2 \cdot f(c) - 1f(3 \cdot c) = 4 \cdot f(c) - 3 \cdot f(c) = 1 \cdot f(c)$. However, $m^{47.2}$ is reducible as $m^{47.2} = m^{47.3} + m^{47.4}$. Moreover, we observe that $m^{47.3}$ and $m^{47.4}$ are irreducible.

In Figure 47d, the algebraic inequality I_{47} over the distributed data scheme $(P_{47}, F_{47}, \equiv_\emptyset)$ is shown. The right-hand side of I_{47} is negative. The marking $m^{47.5} = 0$ is an irreducible solution and an irreducible zero, as $0 \geqq_\emptyset -3 \cdot c$. The marking $m^{47.6}$ is neither a solution nor a zero of I_{47}, as $-4 \cdot c \not\geqq_\emptyset -3 \cdot c$. In contrast to that, $m^{47.7}$ is a solution. However, $m^{47.7}$ is reducible as $m^{47.7} = m^{47.8} + m^{47.9}$, $m^{47.8}$ is a solution, and $m^{47.9}$ is a zero. Moreover, $m^{47.8}$ is an irreducible solution, and $m^{47.9}$ an irreducible zero. The last marking, $m^{47.9}$ is not a solution, but an irreducible zero. Although $0 < m^{47.8} < m^{47.9}$, $m^{47.9}$ is not reducible. The reason is that $m^{47.9} - m^{47.8}$ is not a zero.

places P_{47}: A, B

function symbols F_{47}: $f : S \to S, c :\to S$

(a) The distributed data scheme $(P_{47}, F_{47}, \equiv_\emptyset)$ for the examples.

$$2 \cdot \mathbf{A} - 1 \cdot f(\mathbf{B}) \doteq 1 \cdot f(c)$$

(b) Algebraic equation E_{47}.

	A	B	solution?	zero?	irreducible?
$m^{47.1}$	0	0	×	✓	✓
$m^{47.2}$	$2 \cdot f(c)$	$3 \cdot c$	✓	×	×
$m^{47.3}$	$1 \cdot f(c)$	$1 \cdot c$	✓	×	✓
$m^{47.4}$	$1 \cdot f(c)$	$2 \cdot c$	×	✓	✓

(c) Solutions and zeros of E_{47}.

$$-2 \cdot \mathbf{A} + 1 \cdot f(\mathbf{B}) \dot{\geqq} -3 \cdot c$$

(d) Algebraic Inequality I_{47}.

	A	B	solution?	zero?	irreducible?
$m^{47.5}$	0	0	✓	✓	✓
$m^{47.6}$	$2 \cdot c$	0	×	×	-
$m^{47.7}$	$1 \cdot c + 1 \cdot f(c)$	$2 \cdot c$	✓	×	×
$m^{47.8}$	$1 \cdot f(c)$	$2 \cdot c$	✓	×	✓
$m^{47.9}$	0	$2 \cdot f(c)$	✓	✓	×

(e) Solutions and zeros of I_{47}.

Figure 47.: Examples of solutions and decompositions in minimal solutions and zeros.

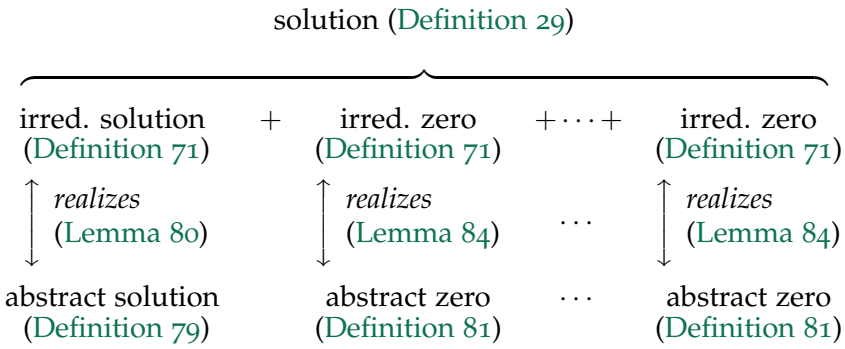


Figure 48.: Representing solutions by abstract solutions and abstract zeros

Lemma 72 gives a first characterization of $\mathbb{L}(\Phi)$. Intuitively, if we take an irreducible solution and add a finite sum of irreducible zeros, we end up with a solution. Moreover, every solution is a sum of an irreducible solution and a finite set of irreducible zeros. Or, put differently, Lemma 72 is equivalent to the following statement:

$$\mathbb{L}(\Phi) = \left\{ s + z^1 + \cdots + z^n \mid s \in \mathbb{L}^{\mathrm{IR}}(\Phi), z^1, \ldots, z^n \in \mathbb{L}_0^{\mathrm{IR}}(\Phi) \right\}$$

Figure 48 shows an overview of our approach to represent solutions: Each solution of an algebraic equation or inequality is a sum of an irreducible solution and finitely many irreducible zeros. This is stated by Lemma 72 and illustrated by the first two lines of Figure 48.

REPRESENTING MARKINGS AS P-VECTORS OF NATURAL NUMBERS. In general both the set of irreducible solutions and the set of irreducible zeros are infinite. Hence, we consider a higher level of abstraction by P-vectors of natural numbers.

First, we define a marking m as θ-unifying, if for all places p and terms of the support of m_p applied to the function $\mathbf{o}_p$ results in a term equivalent to θ.

> **Definition 73 (k-Unifying over θ)**
> Let $\theta \in \langle \frac{\mathbb{V}}{F} \rangle$. Let $a \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ such that for all $p \in P$:
>
> $$\theta' \in \mathrm{support}(a_p) \text{ implies } \mathbf{o}_p(\theta') \equiv \theta$$
>
> Then, a is **k**-unifying over θ.

Then, each P-vector of natural numbers and a term induce a set of **k**-unifying markings as their set of *realizations*.

Definition 74 (Realization of a Vector of Natural Numbers)
Let $\nu \in \mathbb{N}^P$. Let $\theta \in \langle \begin{smallmatrix} \mathbb{V} \\ F \end{smallmatrix} \rangle$. Let $a \in \mathbb{N}\langle \begin{smallmatrix} \mathbb{V} \\ F \end{smallmatrix} \rangle^P$ be $\mathbf{k}$-unifying over θ and for all $p \in P$:

$$\left\| a_p \right\| = \nu_p$$

Then, a is a *realization of* ν *over* θ.

We denote the *set of all realizations of* ν *over* θ by $\mathfrak{R}_\theta(\nu)$.

OUTLOOK. Based on Definition 74 we will represent sets of markings in Sections 9.2 and 9.3. More specifically, we will define *abstract solutions* and *abstract zeros*, which are sets of vectors of natural numbers, whose realizations are irreducible solutions and irreducible zeros, respectively.

In Section 9.2, we will show that each irreducible solution is a realization of an abstract solution and each realization of an abstract solution is an irreducible solution as stated by Lemma 80. This is reflected in the left column of Figure 48. The overview in Figure 48 simplifies the following: Lemma 80 is restricted to the case that r is monomial. However, we show in Section 9.2 that this restriction does not limit our analysis results.

In Section 9.3, we will show that each irreducible zero is a realization of an abstract zero and each realization of an abstract zero is an irreducible zero as stated by lemma Lemma 84. This is reflected in the right columns of Figure 48.

9.2 ABSTRACT SOLUTIONS

In this section, we represent the set of irreducible solutions by a set of *abstract solutions*. First, we show that for every term from $\text{support}(r)$, it is sufficient to consider the respective monomial.

Then, we may assume that r is monomial for the rest of this section. We show that realizations of abstract solutions characterize the set of irreducible solutions.

We assume $r \neq 0$ in this section. The case $r = 0$ is considered in the following section, when characterizing irreducible zeros.

9.2.1 Monomial Right-Hand Side

In this section, we show that is sufficient to consider monomial right-hand side. We reason in two steps:

1. We show that we may assume that no terms in the support of the right-hand side r are equivalent without loss of generality. That is, for all $\theta, \theta' \in \text{support}(r)$: $\theta \neq \theta'$ implies $\theta \not\equiv \theta'$.

2. In general, the right-hand side r can be seen as a sum of monomials $k_1 \cdot \omega_1 + \cdots + k_n \cdot \omega_n$. Considering each monomial separately as right-hand side, we obtain a new algebraic equation or inequality Φ_i for each $1 \leqslant i \leqslant n$. The sum of one irreducible solution for each Φ_i yields an irreducible solution of Φ.

NON-EQUIVALENT SUPPORT OF r. This paragraph is motivated by the following observation. Let θ and θ' be two equivalent terms. Then, the following polynomials are also equivalent:

$$2 \cdot \theta + 3 \cdot \theta' \equiv 5 \cdot \theta$$

Here, the left-hand side is a polynomial, where the support contains two different, yet equivalent terms. The right-hand side is a polynomial with only a single term.

Intuitively, equivalent polynomials as right-hand side yield the same set of solutions:

Corollary 75 (Right-Hand Side Equivalence)
Let $r' \in \mathbb{Z}\langle \frac{\emptyset}{F} \rangle$ with $r' \equiv r$. Let $\sim \in \{\doteq, \dot{\geqq}\}$.
Then, the following identity holds:

$$\mathbb{L}(\mathbf{k} \odot P \sim r) = \mathbb{L}(\mathbf{k} \odot P \sim r')$$

Proof of Corollary 75. Follows from the transitivity of $\equiv$.

Hence, without loss of generality, we assume that for all $\theta, \theta' \in r$, $\theta \neq \theta'$ implies $\theta \not\equiv \theta'$.

MONOMIAL r. In this paragraph we characterize irreducible solutions of algebraic equations and inequalities where the right-hand side r is monomial.

In general, r has the support $\{\omega_1, \ldots, \omega_n\} \subseteq \langle \frac{\emptyset}{F} \rangle$ for an $n \in \mathbb{N} \setminus \{0\}$ and ground terms $\{\omega_1, \ldots, \omega_n\} \subseteq \langle \frac{\emptyset}{F} \rangle$. Accordingly, r is also a finite sum of monomials

$$r = k_1 \cdot \omega_1 + \cdots + k_n \cdot \omega_n$$

for integers $\{k_1, \ldots, k_n\} \subset \mathbb{Z} \setminus \{0\}$. For $i \in \{1, \ldots, n\}$, each monomial $k_i \cdot \omega_i$ induces also an equation $\mathbf{k} \odot P \doteq k_i \cdot \omega_i$ and inequality

$\mathbf{k} \odot P \dot{\geqq} k_i \cdot \omega_i$,respectively. In the following we will show that summing irreducible solutions of each smaller induced algebraic equation or inequality yields an irreducible solution of E or I. And symmetrically, we show that every irreducible solution of E or I is a sum of irreducible solutions the algebraic equations and inequalities with monomial right-hand side.

As an auxiliary lemma, we show that all terms of a support of an irreducible marking result in a term of the support of r, when the abstraction query is applied.

Lemma 76 (Resulting Terms of Irreducible Solutions)
Let $m \in \mathbb{L}^{IR}(\Phi)$. Let $p \in P$, $\theta \in \text{support}(m_p)$
Then, there exists $\omega \in \text{support}(r)$ such that:

$$\mathbf{o}_p(\theta) \equiv \omega$$

Proof of Lemma 76. This proof is indirect and we assume the opposite: Let there exist $\hat{p} \in P$, $\hat{\theta} \in \text{support}(m_p)$ with:

$$\mathbf{o}_{\hat{p}}(\hat{\theta}) \not\equiv \omega \qquad (*)$$

Then, we define $\hat{m} \in \mathbb{N}^{\langle \emptyset_F \rangle^P}$ for $p \in P$ and $\theta \in \langle \emptyset_F \rangle$ by:

$$\hat{m}_p(\theta) = \begin{cases} m_p(\theta) & \text{, if } \mathbf{o}_p(\theta) \equiv \mathbf{o}_{\hat{p}}(\hat{\theta}) \\ 0 & \text{, otherwise.} \end{cases}$$

It follows that $0 < \hat{m} < m$. Moreover, for $\theta \in \langle \emptyset_F \rangle$ with $\theta \not\equiv \mathbf{o}_{\hat{p}}(\hat{\theta})$ holds that:

$$(\mathbf{k} \odot \hat{m})(\theta) = 0$$

On the other hand, we have:

$$\sum_{\substack{\theta \in \langle \emptyset_F \rangle \\ \theta \equiv \mathbf{o}_{\hat{p}}(\hat{\theta})}} \mathbf{k} \odot \hat{m}(\theta) = \sum_{\substack{\theta \in \langle \emptyset_F \rangle \\ \theta \equiv \mathbf{o}_{\hat{p}}(\hat{\theta})}} \mathbf{k} \odot m(\theta) = 0 \quad (\text{or } \geqslant 0, \text{ if } \Phi = I)$$

Hence, $\mathbf{k} \odot \hat{m}$ is a zero. Let $m' = m - \hat{m}$. Then, we have:

$$\sum_{\substack{\theta \in \langle \emptyset_F \rangle \\ \theta \equiv \mathbf{o}_{\hat{p}}(\hat{\theta})}} \mathbf{k} \odot m'(\theta) = \sum_{\substack{\theta \in \langle \emptyset_F \rangle \\ \theta \equiv \mathbf{o}_{\hat{p}}(\hat{\theta})}} \mathbf{k} \odot (m - m)(\theta) = 0 = \sum_{\substack{\theta \in \langle \emptyset_F \rangle \\ \theta \equiv \mathbf{o}_{\hat{p}}(\hat{\theta})}} r(\theta)$$

On the other hand, we have for $\theta \in \langle \begin{smallmatrix} \emptyset \\ F \end{smallmatrix} \rangle$ with $\theta \not\equiv \mathbf{o}_{\hat{P}}(\hat{\theta})$:

$$\sum_{\substack{\theta \in \langle \begin{smallmatrix} \emptyset \\ F \end{smallmatrix} \rangle \\ \theta' \equiv \mathbf{o}_{\hat{P}}(\hat{\theta})}} \mathbf{k} \odot m'(\theta) = \sum_{\substack{\theta \in \langle \begin{smallmatrix} \emptyset \\ F \end{smallmatrix} \rangle \\ \theta' \equiv \mathbf{o}_{\hat{P}}(\hat{\theta})}} \mathbf{k} \odot m(\theta)$$

As m is a solution, m' is also a solution. Thus m is reducible with $\hat{m}$ and m', which is a contradiction. Hence, we conclude that $*$ is wrong.

Lemma 77 states that it is possible to decompose an irreducible solution into irreducible solutions of the respective decomposed equations or inequalities, respectively.

Lemma 77 (Reducing Support of Right-Hand Side)
Let $|\text{support}(r)| \geqslant 2$. Let $\omega \in \text{support}(r)$. Let $r_\omega = r(\omega) \cdot \omega$. Let $r' = r - r_\omega$. Let $\sim \in \{\doteq, \dot{\geqq}\}$. Let $m \in \mathbb{N}\langle \begin{smallmatrix} \emptyset \\ F \end{smallmatrix} \rangle^P$.
Then, the following statements are equivalent:

1. $m \in \mathbb{L}^{IR}(\mathbf{k} \odot P \sim r)$.
2. There exist $m^\omega, m' \in \mathbb{N}\langle \begin{smallmatrix} \emptyset \\ F \end{smallmatrix} \rangle^P$ with
   a) $m^\omega \in \mathbb{L}^{IR}(\mathbf{k} \odot P \sim r_\omega)$,
   b) $m' \in \mathbb{L}^{IR}(\mathbf{k} \odot P \sim r')$, and
   c) $m = m^\omega + m'$.

Due to its length, the proof of Lemma 77 can be found at the end of this section.

Now, we apply Lemma 77 inductively and show that without losing generality, we may assume a monomial right-hand side. A finite representation of the set of solutions of each respective algebraic equation or inequality with monomial right-hand side yields a finite representation of all irreducible solutions.

Corollary 78 (Reduction to Monomial right-hand side)
For each $\omega \in \text{support}(r)$, Let $R_\omega \in \mathbb{N}\langle \begin{smallmatrix} \mathbb{V} \\ F \end{smallmatrix} \rangle^P$ be a representation of all irreducible solutions of $\mathbf{k} \odot P \doteq r(\omega) \cdot \omega$. Moreover, let

$$R = \sum_{\omega \in \text{support}(r)} R_\omega$$

Then, R is a representation of all irreducible solutions of $\mathbf{k} \odot P \doteq r$.

$$
\begin{aligned}
&\mathsf{E}_{49.1}: && 1\cdot\mathbf{A}+1\cdot\mathsf{f}(\mathbf{B})=1\cdot\mathsf{c}+1\cdot\mathsf{f}(\mathsf{c})\\
&\mathsf{E}_{49.2}: && 1\cdot\mathbf{A}+1\cdot\mathsf{f}(\mathbf{B})=1\cdot\mathsf{c}\\
&\mathsf{E}_{49.3}: && 1\cdot\mathbf{A}+1\cdot\mathsf{f}(\mathbf{B})=1\cdot\mathsf{f}(\mathsf{c})
\end{aligned}
$$

(a) Decomposition of right-hand side of an algebraic equation.

place p:	A	B	$\in \mathbb{L}^{\mathrm{IR}}(\mathsf{E}_{49.1})$	$\in \mathbb{L}^{\mathrm{IR}}(\mathsf{E}_{49.2})$	$\in \mathbb{L}^{\mathrm{IR}}(\mathsf{E}_{49.3})$
$\mathsf{m}_{\mathsf{p}}^{49.1}$	$1\cdot\mathsf{f}(\mathsf{c})$	0	×	×	✓
$\mathsf{m}_{\mathsf{p}}^{49.2}$	0	$1\cdot\mathsf{c}$	×	×	✓
$\mathsf{m}_{\mathsf{p}}^{49.3}$	$1\cdot\mathsf{c}$	0	×	✓	×
$\mathsf{m}_{\mathsf{p}}^{49.4}$	$1\cdot\mathsf{f}(\mathsf{c})+1\cdot\mathsf{c}$	0	✓	×	×
$\mathsf{m}_{\mathsf{p}}^{49.5}$	$1\cdot\mathsf{c}$	$1\cdot\mathsf{c}$	✓	×	×

(b) Irreducible solutions of the three algebraic equations.

Figure 49.: Examples of decomposition of non-monomial right side

> *Proof of Corollary 78.* We apply Lemma 77 inductively: This is possible as r has finite support and $r - r(\omega)\cdot\omega < r$.

Figure 49 illustrates Lemma 77 and Corollary 78. We use the the distributed data scheme $(\mathsf{P}_{47}, \mathsf{F}_{47}, \equiv_\emptyset)$ from Figure 47a with two places A and B, and symbols c and f. We consider the three algebraic equations shown in Figure 49a. The abstraction query is equal for all algebraic equations. The right-hand side of $\mathsf{E}_{49.1}$, $1\cdot\mathsf{c}+1\cdot\mathsf{f}(\mathsf{c})$ is the sum of the monomial right-hand side of $\mathsf{E}_{49.2}$ and $\mathsf{E}_{49.3}$. Figure 49b shows irreducible solutions of all three algebraic equations. $\mathsf{m}^{49.1}$ and $\mathsf{m}^{49.2}$ are irreducible solutions of $\mathsf{E}_{49.3}$. $\mathsf{m}^{49.3}$ is an irreducible solution of $\mathsf{E}_{49.2}$. $\mathsf{m}^{49.4}$ and $\mathsf{m}^{49.5}$ are irreducible solutions of $\mathsf{E}_{49.1}$. Moreover, the following equations hold:

$$
\begin{aligned}
\mathsf{m}^{49.4} &= \mathsf{m}^{49.1} + \mathsf{m}^{49.3}\\
\mathsf{m}^{49.5} &= \mathsf{m}^{49.2} + \mathsf{m}^{49.3}
\end{aligned}
$$

Thus, summing the irreducible solutions of the equations $\mathsf{E}_{49.2}$ and $\mathsf{E}_{49.3}$ with monomial right-hand side results in an irreducible solution of $\mathsf{E}_{49.1}$.

9.2.2 Abstract Solutions for Monomial Right-Hand Side

In this section, we show that *abstract solutions* characterize the set of irreducible solutions, if the right-hand side r is monomial.

As a result of Corollary 78, we make the following assumption without loss of generality: r is monomial. Thus, we fix $\omega \in \langle {}^{\emptyset}_{F} \rangle$ such that:

$$\{\omega\} = \text{support}(r)$$

In the following, we define *abstract solutions*. Intuitively, E induces a linear Diophantine equation, and I induces a linear Diophantine inequality, as defined in Definition 60 (Page 129). Then, each irreducible solution $\nu \in \mathbb{N}^P$ of the induced linear Diophantine equation or inequality, is an abstract solution. Then, we consider all realizations resulting in **k**-unifying markings over ω.

> **Definition 79** (Abstract Solution)
> Let $\{x_p \mid p \in P\}$ be a set of unknowns. Let E^D be the Diophantine equation $\sum_{p \in P} \ell_p x_p \doteq r(\omega)$. Let I^D be the Diophantine inequality $\sum_{p \in P} \ell_p x_p \dot{\geqslant} r(\omega)$. Let $\nu \in \mathbb{N}^P$ be an irreducible solution of Φ^D.
>
> Then, ν is an abstract solution of Φ.

We say m *realizes* ν, or: m *is a realization of* ν, if $m \in \mathfrak{R}_\omega(\nu)$.

Figure 50 shows some examples of abstract solutions and their realizations. Figure 50a and Figure 50b show the algebraic inequality I_{50} and the induced linear Diophantine inequality I^D_{50}, respectively. ν^1, ν^2, and ν^3 are abstract solutions of I_{50}, as they are irreducible solutions of I^D_{50}. As $f(c)$ is the only member of the support, we consider all realizations which are unifying markings over $f(c)$. Here, the marking $m^{50.1}$ is unifying over $f(c)$. The reason is that both $f(c)$ applied to $\mathbf{A}$ and c applied to $f(\mathbf{B})$ result in $f(c)$. Moreover, we have $\left\| m^{50.1}_A \right\| = \nu^1_A = 2$ and $\left\| m^{50.1}_B \right\| = \nu^1_B = 0$. We observe that $m^{50.1}$ is the only realization over $f(c)$ of ν^1. The markings $m^{50.2}$ and $m^{50.3}$ are realizations of ν^2 and ν^3, respectively.

We observe the following: Let ν be an abstraction solution and $p \in P$ with $\nu_p > 0$ and $\mathbf{o}_p$ is not unifiable with ω. Then, the set of realizations of ν is empty. On the other hand, if $\mathbf{o}_p$ is a ground term, the set of realizations of ν is infinite.

In the following we prove the main result of this section: Lemma 80, which is visualized as a Venn diagram by Figure 51. We show that every realization of an abstract solution is an *irreducible* solution. Symmetrically, every irreducible solution is a realization of an abstract solution.

$$\mathsf{I}_{50}: \qquad 2 \cdot \mathbf{A} + 1 \cdot \mathsf{f}(\mathbf{B}) \dot{\geqq} 3 \cdot \mathsf{f}(\mathsf{c})$$

(a) The algebraic inequality I_{50} over the distributed data scheme $(\mathsf{P}_{47}, \mathsf{F}_{47}, \equiv_{\emptyset})$.

$$\mathsf{I}_{50}^{\mathsf{D}}: \qquad 2x_{\mathsf{A}} + 1x_{\mathsf{B}} \dot{\geqslant} 3x$$

(b) The linear Diophantine inequality $\mathsf{I}_{50}^{\mathsf{D}}$ induced by I_{50} with unknowns x_{A} and x_{B}.

place p:	A	B	
ν_{p}^{1}:	2	0	abstract solution
ν_{p}^{2}:	1	1	abstract solution
ν_{p}^{3}:	0	3	abstract solution
$\mathsf{m}^{50.1}$	$2 \cdot \mathsf{f}(\mathsf{c})$	0	$\in \mathfrak{R}_{\mathsf{f}(\mathsf{c})}\left(\nu^{1}\right)$
$\mathsf{m}^{50.2}$	$1 \cdot \mathsf{f}(\mathsf{c})$	$1 \cdot \mathsf{c}$	$\in \mathfrak{R}_{\mathsf{f}(\mathsf{c})}\left(\nu^{2}\right)$
$\mathsf{m}^{50.3}$	0	$3 \cdot \mathsf{c}$	$\in \mathfrak{R}_{\mathsf{f}(\mathsf{c})}\left(\nu^{3}\right)$

(c) Three abstract solutions with three realizations.

Figure 50.: Abstract solutions and abstract zeros of an algebraic inequality.

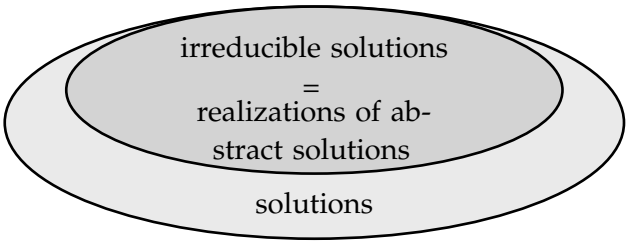


Figure 51.: Venn diagram of solutions, realizations of abstract markings, and irreducible solutions assuming r is monomial.

Lemma 80 (Realizations of Abstract Solutions are Irreducible Solutions)
Let A be the set of all abstract solutions of Φ. Then, the following holds:

$$\mathbb{L}^{\mathrm{IR}}(\Phi) = \bigcup_{\nu \in A} \mathfrak{R}_\omega(\nu)$$

Proof of Lemma 80. "$\subseteq$": Let $m \in \mathbb{L}^{\mathrm{IR}}(\Phi)$. We define $\nu \in \mathbb{N}^P$ for $p \in P$ by: $\nu_p = \|m_p\|$. By Lemma 76, we deduce that for all $p \in P$, $\theta \in \mathrm{support}(m_p)$:

$$\mathbf{o}_p(\theta) \equiv \omega$$

It remains to show that $\nu \in \mathbb{N}^P$ with $\nu_p = \|m_p\|$ is an irreducible solution for the linear Diophantine equation $\sum_{p \in P} \ell_p x_p = r(\omega)$ or linear Diophantine inequality $\sum_{p \in P} \ell_p x_p \geqslant r(\omega)$, respectively. We first observe that ν is a solution:

$$\begin{aligned}\sum_{p \in P} \ell_p \|m_p\| &= \sum_{p \in P} \ell_p \|\mathbf{o}_p(m_p)\| \\ &= \sum_{p \in P} \ell_p \left(\mathbf{o}_p(m_p)\right)(\omega) \\ &= \mathbf{k} \odot m(\omega)\end{aligned}$$

And hence, $\sum_{p \in P} \ell_p \|m_p\| = r(\omega)$ or $\sum_{p \in P} \ell_p \|m_p\| \geqslant r(\omega)$, respectively. It remains to show that ν is an *irreducible* solution. We show this indirect and assume the opposite:

$$\nu \text{ is reducible} \tag{$*$}$$

Then, there exist a solution $\nu' < \nu$ and a zero $\nu'' < \nu$ of the induced linear Diophantine equation or inequality with $\nu = \nu' + \nu''$. Then, there exist markings $m', z \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$ with $0 < m' < m$, $0 < z < m$, and $m = m' + z$, and $\|m_p\| = \nu'_p$ and $\|z_p\| = \nu''_p$ for all $p \in P$.

As $m' < m$, it follows that m' is also $\mathbf{k}$-unifying over ω and thus: $\mathbf{k} \odot m' = \sum_{p \in P} \ell \left\| m'_p \right\| \cdot \omega = \sum_{p \in P} \ell \nu'_p$. It follows that m' is a solution. Equivalently, $\mathbf{k} \odot z' = \sum_{p \in P} \ell \left\| z'_p \right\| \cdot \omega = \sum_{p \in P} \ell x''_p$ and thus z is a zero. Hence, m is reducible with the solution m' and the zero z. This is a contradiction to the assumption that m is irreducible. Hence, $*$ is wrong. We derive that ν is an

irreducible solution if the induced linear Diophantine equation or inequality, respectively.
"$\supseteq$": Let $\nu \in A$ be an abstract solutions of Φ and $m \in \mathfrak{R}_\omega(\nu)$. By Definition 74 we have for all $p \in P$, $\theta \in \text{support}(m_p)$ that:

$$\mathbf{o}_p(\theta) \equiv \omega \tag{$**$}$$

Moreover, as ν is a solution to the induced linear Diophantine equation or inequality, one of the following holds:

$$\sum_{p \in P} \ell_p \left\| m_p \right\| = r(\omega) \tag{$***$}$$

$$\sum_{p \in P} \ell_p \left\| m_p \right\| \geqslant r(\omega) \tag{$****$}$$

Thus, we have by by $**$:

$$\mathbf{k} \odot m = \sum_{p \in P} \ell_p \mathbf{o}_p(m_p) \equiv \sum_{p \in P} \ell_p \left\| m_p \right\| \cdot \omega$$

If ν is an abstract solution of E, we deduce by $***$:

$$\mathbf{k} \odot m \equiv r(\omega) \cdot \omega$$

Otherwise, ν is an abstract solution of I, and we deduce by $****$:

$$\mathbf{k} \odot m \geqq r(\omega) \cdot \omega$$

Hence, m is a solution of E (I).

It remains to show that m is irreducible, which we show indirect:

Assumption: m is reducible.

Then, there exist $0 < m' < m$ and $0 < z < m$ where m' is a solution and z is a zero and $m = m' + z$. As m is $\mathbf{k}$-unifying over ω, we have:

$$\mathbf{k} \odot m' = \sum_{p \in P} \ell_p \left\| m'_p \right\| \cdot \omega$$

and

$$\mathbf{k} \odot z = \sum_{p \in P} \ell_p \left\| z_p \right\| \cdot \omega$$

Thus, $\nu', \nu'' \in \mathbb{N}^P$ with $\nu'_p = \left\| m'_p \right\|$ and $\nu'_p = \left\| z_p \right\|$ are a solution and zero of the respection linear Diophantine equation or inequality. Hence, ν is reducible, which is a contradiction.

Hence, the set of abstract solutions induces the set of all irreducible solutions, if r is monomial.

Finally, we present the missing proof of Lemma 77 to complete the section on abstract solutions.

Proof of Lemma 77. 1.⇒2.: We define m^ω as follows for $p \in P$:

$$m^\omega_p = \begin{cases} m_p(\theta) & \text{, if } \mathbf{o}_p(\theta) \equiv \omega \\ 0 & \text{, otherwise.} \end{cases}$$

Then, it holds that:

$$\sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \omega}} (\mathbf{k} \odot m^\omega)(\theta) = \sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \omega}} (\mathbf{k} \odot m)(\theta) = r(\omega) \quad \text{or } (\geqslant r(\omega))$$

and on the other hand that for all $\theta \in \langle \frac{\emptyset}{F} \rangle$ with $\theta \not\equiv \omega$:

$$\mathbf{k} \odot m^\omega(\theta) = 0$$

Hence, we deduce that $\mathbf{k} \odot m^\omega \sim r(\omega) \cdot \omega$. Hence, m^ω is a solution of $\mathbf{k} \odot P \sim r(\omega) \cdot \omega$. It remains to show that m^ω is irreducible. Assume the opposite: Then there would exist a zero $z < m^\omega \leqslant m$, and then $m - z$ is also solution of $\mathbf{k} \odot P \sim r$. As the zero of both equations (and inequalities) coincide, m would be reducible.

Accordingly, we define $m' = m - m^\omega$ and observe that m' is a solution of $\mathbf{k} \odot P \sim r - r(\omega) \cdot \omega$. It remains to show that m' is an irreducible solution. Assume the opposite: Then there would exist a zero $z < m' \leqslant m$, and then m would be also reducible.

2.⇒1.: We apply Lemma 76: As m' is irreducible, we have for all $p \in P$:

$$\theta \in \text{support}(m'_p) \text{ implies } \mathbf{o}_p(\theta) \not\equiv \omega\,. \tag{$*$}$$

Moreover, as m^ω is irreducible, we have:

$$\theta \in \text{support}(m^\omega_p) \text{ implies } \mathbf{o}_p(\theta) \equiv \omega\,. \tag{$**$}$$

We observe for $m = m' + m^\omega$:

$$\begin{aligned} \mathbf{k} \odot m &= \mathbf{k} \odot (m_\omega + m) \\ &= \mathbf{k} \odot m_\omega + \mathbf{k} \odot m' \\ &\sim r(\omega) \cdot \omega + r - r(\omega) \cdot \omega \end{aligned}$$

$$= r$$

Hence, m is a solution. It remains to show that m is irreducible. We show this property indirect by making the following assumption:

$$\textit{Assumption: } m \text{ is reducible.} \qquad (***)$$

Then, there exist $0 < \underline{m} < m$, $0 < z < m$ where $\underline{m}$ is a solution, z is a zero and $m = \underline{m} + z$. Moreover z is a zero of $\mathbf{k} \odot P \sim r(\omega) \cdot \omega$ and $\mathbf{k} \odot P \sim r - r(\omega) \cdot \omega$.

Let $z^\omega \in \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$ defined for all $p \in P$, $\theta \in \langle \frac{\emptyset}{F} \rangle$ by:

$$z_p^\omega(\theta) = \begin{cases} z_p(\theta) & \text{, if } \mathbf{o}(\theta) \equiv \omega \\ 0 & \text{, otherwise.} \end{cases} \qquad (****)$$

and $z' = z - z^\omega$. We observe that $z^\omega < m^\omega$ and $z' < m'$. Then, we have:

$$\sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \omega}} (\mathbf{k} \odot z^\omega)(\theta) = \sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \omega}} (\mathbf{k} \odot z)(\theta).$$

and

$$\sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \not\equiv \omega}} (\mathbf{k} \odot z')(\theta) = \sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \not\equiv \omega}} (\mathbf{k} \odot z)(\theta).$$

Moreover, for all $\theta \in \langle \frac{\emptyset}{F} \rangle$ with $\theta \not\equiv \omega$: $\mathbf{k} \odot z^\omega = 0$ and for all $\theta \equiv \omega$: $\mathbf{k} \odot z' = 0$. Hence, we deduce z^ω and z' are zeros of all three equations, or inequalities respectively. Furthermore let $\underline{m}^\omega \in \mathbb{N}\langle \frac{\emptyset}{F} \rangle^P$ defined by:

$$\underline{m}_p^\omega(\theta) = \begin{cases} \underline{m}_p(\theta) & \text{, if } \mathbf{o}_p(\theta) \equiv \omega \\ 0 & \text{, otherwise.} \end{cases}$$

and $\underline{m}' = \underline{m} - \underline{m}^\omega$. Then, we have $\underline{m}' + z' = m'$ and $\underline{m}^\omega + z^\omega = m^\omega$. We distinguish the following cases.

1st case: $z^\omega = 0$.
Then, $z' = z - z^\omega = z$. In this case, we will show that m' is a reducible solution of $\mathbf{k} \odot P \sim r - r(\omega) \cdot \omega$. As $z' > 0$, it remains to show that $\underline{m}'$ is a solution of $\mathbf{k} \odot P \sim r - r(\omega) \cdot \omega$. For $\theta \in \langle \frac{\emptyset}{F} \rangle$ with $\theta \equiv \omega$, we have $\mathbf{k} \odot \underline{m}' = 0 = \big(r - r(\omega) \cdot \omega\big)(\omega) = \big(r - r(\omega) \cdot \omega\big)(\theta)$, as $\theta \equiv \omega$ and

$\theta \in \text{support}(r)$ implies $\theta = \omega$. For $\theta \in \langle {}^{\emptyset}_{F} \rangle$ with $\theta \not\equiv \omega$ it holds that:

$$
\begin{aligned}
& \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} \left(\mathbf{k} \odot (\underline{m}')\right)(\theta') \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} \left(\mathbf{k} \odot (m' - z)\right)(\theta') \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} \left(\mathbf{k} \odot (m' - z)\right)(\theta') + 0 \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} \left(\mathbf{k} \odot (m' - z)\right)(\theta') + (\mathbf{k} \odot m^{\omega})(\theta') \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} \left(\mathbf{k} \odot (m' + m^{\omega} - z)\right)(\theta) \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} \left(\mathbf{k} \odot (m - z)\right)(\theta') \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} (\mathbf{k} \odot \underline{m})(\theta') \\
\underset{(\text{or } =)}{\geqslant} & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} r(\theta') \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} r(\theta') - 0 \\
= & \sum_{\substack{\theta' \in \langle {}^{\emptyset}_{F} \rangle \\ \theta' \equiv \theta}} r(\theta') - r(\omega) \cdot \omega
\end{aligned}
$$

Thus $\underline{m}'$ is a solution of $\mathbf{k} \odot P \sim r - r(\omega) \cdot \omega$. Thus, m' is a reducible solution of $\mathbf{k} \odot P \sim r - r(\omega) \cdot \omega$, which is a contradiction to the prerequisite.

2^{nd} case: $z^{\omega} > 0$.
By $*$ and $**$, it follows that $0 < z^{\omega} \leqslant m^{\omega}$. In the rest of this case, we will show that $\underline{m} = m^{\omega} - z^{\omega}$ is a solution

of $\mathbf{k} \odot P \sim r(\omega) \cdot \omega$, which implies that m^ω is reducible, which is a contradiction. First, by $**$, it follows that for all $\theta \in \langle \frac{\emptyset}{F} \rangle$ with $\theta \not\equiv \omega$: $\mathbf{k} \odot m^\omega = \mathbf{k} \odot z^\omega = r(\omega) \cdot \omega(\theta) = 0$. Then, we have:

$$
\begin{aligned}
&\sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} \left(\mathbf{k} \odot (m^\omega - z^\omega)\right)(\theta') \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} \left(\mathbf{k} \odot (m^\omega - z^\omega)\right)(\theta') + 0 - 0 \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} \left(\mathbf{k} \odot (m^\omega - z^\omega)\right)(\theta') + \left(\mathbf{k} \odot m'\right)(\theta') + \left(\mathbf{k} \odot z'\right)(\theta') \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} \left(\mathbf{k} \odot (m^\omega + m' - z^\omega - z')\right)(\theta') \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} \left(\mathbf{k} \odot (m - z)\right)(\theta') \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} (\mathbf{k} \odot \underline{m})(\theta') \\
&\underset{\text{(or =)}}{\geqslant} \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} r(\theta') \\
&= \sum_{\substack{\theta' \in \langle \frac{\emptyset}{F} \rangle \\ \theta' \equiv \omega}} \left(r(\omega) \cdot \omega\right)(\theta')
\end{aligned}
$$

Hence, we deduce that $\mathbf{k} \odot (m^\omega - z^\omega) \sim r(\omega) \cdot \omega$. Then, $\underline{m}^\omega$ is a solution of $\mathbf{k} \odot P \sim r(\omega) \cdot \omega$. Thus, m^ω is reducible, which is a contradiction to the prerequisite.

Hence, we led $***$ to contradiction and deduce the opposite: m is irreducible.

9.3 ABSTRACT ZEROS

In this section, we characterize the set of irreducible zeros of an algebraic equation or inequality by *abstract zeros*.

$$I_{52}: \qquad 2 \cdot \mathbf{A} + 1 \cdot f(\mathbf{B}) \dot{\geqq} 3 \cdot f(c)$$

(a) The algebraic inequality I_{52} over the distributed data scheme $(P_{47}, F_{47}, \equiv_{\emptyset})$.

$$I_{52}^{D}: \qquad 2x_A + 1x_B \dot{\geqslant} 3x$$

(b) The linear Diophantine inequality I_{52}^{D} induced by I_{52} with unknowns x_A and x_B.

place p:	A	B	
ν_p^4:	1	0	abstract zero
ν_p^5:	0	1	abstract zero
$z^{52.1}$	$1 \cdot c$	0	$\in \mathfrak{R}_c\left(\nu^4\right)$
$z^{52.2}$	0	$1 \cdot c$	$\in \mathfrak{R}_{f(c)}\left(\nu^5\right)$
$z^{52.3}$	0	$1 \cdot f(c)$	$\in \mathfrak{R}_{f(f(c))}\left(\nu^5\right)$

(c) Three abstract zeros with three realizations.

Figure 52.: Abstract zeros of an algebraic inequality.

First, we will show that irreducible zeros are **k**-unifying. When considering irreducible solutions, we considered **k**-unifying solutions over ω, where $\omega \in \text{support}(r)$. In contrast to that irreducible zeros may be **k**-unifying over arbitrary terms.

> **Definition 81** (Abstract Zero)
> Let $\{x_p \mid p \in P\}$ be a set of unknowns. Let E^D be the Diophantine equation $\sum_{p \in P} \ell_p x_p \doteq 0$. Let I^D be the Diophantine inequality $\sum_{p \in P} \ell_p x_p \dot{\geqslant} 0$. Let $\nu \in \mathbb{N}^P$ be an irreducible solution of Φ^D.
>
> Then, ν is an abstract solution of Φ.

For an abstract zero ν and any $\theta \in \left\langle \begin{smallmatrix} \mathbb{V} \\ F \end{smallmatrix} \right\rangle$, we call a marking $m \in \mathfrak{R}_\theta(\nu)$ a *realization of* ν and say m *realizes* ν over θ.

Figure 52 illustrates abstract zeros and their realizations. In Figures 52a and 52b, we restate the algebraic inequality I_{50} and its induced Diophantine inequality I_{50}^{D} from Figure 50 of the previous section, respectively. In Figure 52c, two abstract zeros are shown: ν^4

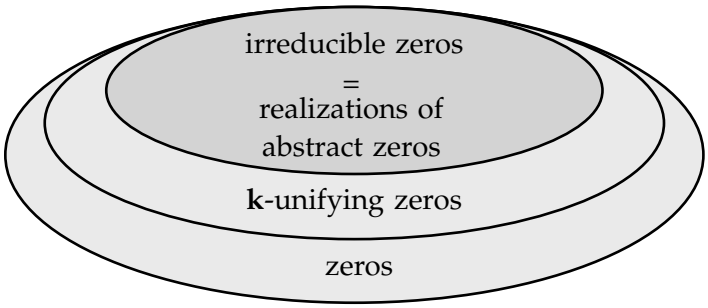


Figure 53.: Venn diagram of realizations of abstract zeros, irreducible zeros, **k**-unifying zeros and zeros.

and ν^5. Moreover, $z^{52.1}$ realizes ν^4, whereas $z^{52.2}$ and $z^{52.3}$ realize ν^5. Accordingly, each of the three markings $z^{52.1}$, $z^{52.2}$, and $z^{52.3}$ is an irreducible zero.

Figure 53 illustrates the relation of realizations of abstract zeros, irreducible zeros, **k**-unifying zeros and zeros. In the rest of this section, we prove the relations shown in the Venn diagram.

ZEROS ARE k-UNIFYING. In this paragraph, we show that every zero is a sum of **k**-unifying zeros. First, we observe that we can decompose every zero into a sum of **k**-unifying zeros.

Lemma 82 (Every Zero is a Sum of **k**-Unifying Zeros)
Let $z \in \mathbb{L}_0(\Phi)$ be a zero of Φ.
Then, there exist $z^1, \ldots, z^n \in \mathbb{L}_0(\Phi)$ with:

1. $z = \sum_{i=1}^{n} z^i$,
2. $z^i > 0$ for all $1 \leqslant i \leqslant n$, and
3. z^i is a **k**-unifying zeros for all $1 \leqslant i \leqslant n$.

Proof of Lemma 82. Let $\hat{p} \in P$ and $\xi \in \operatorname{support}(z_{\hat{p}})$. We define $z' \in \mathbb{N}\langle {}^{\emptyset}_{F} \rangle^P$ as follows for $\theta \in \langle {}^{\emptyset}_{F} \rangle$, $p \in P$:

$$z'_p(\theta) = \begin{cases} z_p(\theta) & \text{, if } \mathbf{o}_p(\theta) \equiv \mathbf{o}_{\hat{p}}(\xi) \\ 0 & \text{, otherwise.} \end{cases}$$

Then, $0 < z' \leqslant z$. By construction, we have for $\theta \in \langle \frac{\emptyset}{F} \rangle$ with $\theta \not\equiv \mathbf{o}_{\hat{p}}(\xi)$ that $\mathbf{k} \odot z'(\theta) = 0$. Moreover, we have:

$$\sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \mathbf{o}_{\hat{p}}(\xi)}} \mathbf{k} \odot z'(\theta) = \sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \mathbf{o}_{\hat{p}}(\xi)}} \mathbf{k} \odot z(\theta)$$

Hence, z' is a zero, as z is a zero of Φ. By construction, we observe that z' is a $\mathbf{k}$-unifying zero over $\mathbf{o}_{\hat{p}}(\xi)$. Let $z'' = z - z'$. Then, we have:

$$\mathbf{k} \odot z''(\theta) = \begin{cases} 0 & \text{, if } \theta \equiv \mathbf{o}_{\hat{p}}(\xi) \\ \mathbf{k} \odot z(\theta) & \text{, otherwise.} \end{cases}$$

Hence, z'' is also a zero. If $z = z'$, then z is a $\mathbf{k}$-unifying zero and the assumption follows. Otherwise, as $z'' < z$ we may apply induction: Then, z'' is a sum of $\mathbf{k}$-unifying zeros $z^1, \ldots, z^n$. Hence, $z = \sum_{i=1}^{n} z^i + z'$ is also a sum of $\mathbf{k}$-unifying zeros.

The lemma implies the following corollary: Each irreducible zero is a $\mathbf{k}$-unifying zero.

Corollary 83 (Every Irreducible Zero is a $\mathbf{k}$-Unifying Zero)
Let $z \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ be an irreducible zero of Φ. Then, z is $\mathbf{k}$-unifying.

REALIZATIONS OF ABSTRACT ZEROS. Now, we can state the main result of this section: The set of abstract zeros of Φ characterizes the set of irreducible zeros of Φ.

Lemma 84 (Realizations of Abstract Zeros are Irreducible Zeros)
Let A be the set of all abstract zeros of Φ. Then, the following holds:

$$\mathbb{L}^{\mathbb{IR}}(\Phi) = \bigcup_{\substack{\nu \in A \\ \theta \in \langle \frac{\emptyset}{F} \rangle}} \mathfrak{R}_\theta(\nu)$$

Proof of Lemma 84. Let D be the induced linear Diophantine equation or inequality of Φ: $\sum_{p \in P} \ell_p x_p = 0$ or $\sum_{p \in P} \ell_p x_p \geqslant 0$, respectively.

"$\subseteq$": Let $z \in \mathbb{L}^{\mathrm{IR}}(\Phi)$. By Lemma 82, z is $\mathbf{k}$-unifying over $\hat{\theta}$ for some $\hat{\theta} \in \langle \frac{\emptyset}{F} \rangle$. We define $\nu \in \mathbb{N}^P$ for $p \in P$ by $\nu_p = \left\| z_p \right\|$. Then, we have:

$$\sum_{p \in P} \ell_p \nu_p = \sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \hat{\theta}}} \mathbf{k} \odot z(\theta) = 0 \quad (\geqslant 0)$$

Thus, ν_p is a zero of D. As z is $\mathbf{k}$-unifying, we have $\mathbf{o}_p(\theta_p) \equiv \hat{\theta}$. Thus, z is a realization of ν.

It remains to show that ν is an irreducible solution of D. Assume the opposite:

$$\textit{Assumption}\text{: } \nu \text{ is a reducible solution of D} \qquad (*)$$

Let $\nu = \nu' + \nu''$ with $\nu' \neq 0$ and $\nu'' \neq 0$, where both, ν' and ν'', are zeros of D. Then, there exists $0 < z' < z$ and $0 < z'' < z$ such that $z' \in \mathfrak{R}_{\hat{\theta}}(\nu')$, $z'' \in \mathfrak{R}_{\hat{\theta}}(\nu'')$ and $z = z' + z''$ by splitting the polynomials for each place. Accordingly, both z' and z'' are $\mathbf{k}$-unifying zeros. Thus, z is reducible, which is a contradiction. Thus, the assumption $*$ is wrong and we deduce that ν is an irreducible zero of D.

"$\supseteq$": Let $\nu \in A$ and z be realization of ν. Then, there exists $\hat{\theta} \in \langle \frac{\emptyset}{F} \rangle$ such that for all $p \in P$ and $\theta \in \mathrm{support}(z_p)$ we have $\mathbf{o}_p(\theta) = \hat{\theta}$. Hence, we have for $\theta \in \langle \frac{\emptyset}{F} \rangle$ with $\theta \not\equiv \hat{\theta}$ that $\mathbf{k} \odot z(\theta) = 0$. On the other hand, we have:

$$\sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \hat{\theta}}} \mathbf{k} \odot z(\theta) = \sum_{p \in P} \ell_p \|z\|_p = \sum_{p \in P} \ell_p \nu_p$$

And as ν is zero of D, we have that $\mathbf{k} \odot z \equiv 0$ or $\mathbf{k} \odot z \geqq 0$, respectively. Thus, z is a zero of Φ.

It remains to show that z is irreducible. We show this indirect by assuming the opposite.

$$\textit{Assumption}\text{: } z \text{ is a reducible zero of } \Phi. \qquad (**)$$

Then, there exist $z', z'' \in \mathbb{L}_0(\Phi) \setminus \{0\}$ such that $z = z' + z''$.

Then, z' and z'' are a $\mathbf{k}$-unifying zeros, as $z' < z$ and $z'' < z$. Let $\nu', \nu'' \in \mathbb{N}^P$ with $\nu'_p = \left\| z'_p \right\|$ and $\nu''_p = \left\| z''_p \right\|$. For $\theta \in \langle \frac{\emptyset}{F} \rangle$

with $\theta \not\equiv \hat{\theta}$, we have $\mathbf{k} \odot z'(\theta) = \mathbf{k} \odot z''(\theta) = 0$. On the other hand, we have:

$$\sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \hat{\theta}}} \mathbf{k} \odot z'(\theta) = \sum_{p \in P} \ell_p \left\| z' \right\|_p = \sum_{p \in P} \ell_p \nu'_p$$

and

$$\sum_{\substack{\theta \in \langle \frac{\emptyset}{F} \rangle \\ \theta \equiv \hat{\theta}}} \mathbf{k} \odot z''(\theta) = \sum_{p \in P} \ell_p \left\| z'' \right\|_p = \sum_{p \in P} \ell_p \nu''_p .$$

Hence, ν' and ν'' are also zeros of D. Thus, ν is reducible, which is a contradiction. We deduce that $**$ is wrong and z is irreducible.

9.4 COMPUTABILITY

In the previous sections, we defined abstract solutions and abstract zeros, which induce irreducible solutions and irreducible zeros. Now, we use abstract solutions and abstract zeros to deduce that a finite representation of all irreducible solutions and all irreducible zeros is computable, if $(F, \equiv)$ is compact.

Observing the structural definition of terms and term congruences, we deduce the following technical corollary.

Corollary 85 (Congruence on Substitutions)
Let $\theta, \theta' \in \langle \frac{\mathbb{V}}{F} \rangle$ with $\theta \equiv \theta'$. Let $\sigma, \sigma' \in \mathbb{V} \overset{F}{\rightsquigarrow} \mathbb{V}$ with $\sigma \equiv \sigma'$.
Then, $\sigma(\theta) \equiv \sigma(\theta')$.

The following lemma states that the order of applying an abstraction query and substitution is equivalent if the variables used by the parameterized marking are not in P. The proof is a straight-forward application of all involved definitions.

Lemma 86 (Order of Abstraction Query and Substitution)
Let $V \subset \mathbb{V} \setminus P$ be finite. Let $\sigma \in V \overset{F}{\rightsquigarrow} V$. Let $a \in \mathbb{N}\langle \frac{V}{F} \rangle^P$.
Then, the following equation holds:

$$\sigma(\mathbf{k} \odot a) = \mathbf{k} \odot \sigma(a)$$

Proof of Lemma 86. Let $p \in P$. As $\mathbb{V}(\mathbf{o}_p) = p$ and $p \notin V$, the following equality holds for all $\zeta \in \langle {}^{V}_{F} \rangle$:

$$\sigma(\mathbf{o}_p(\zeta)) = \mathbf{o}_p(\sigma(\zeta)) \qquad (*)$$

Let $\theta \in \langle {}^{\mathbb{V}}_{F} \rangle$. Then, we have:

$$\begin{aligned}
\sigma\left(\mathbf{k} \odot a\right)(\theta) &= \sigma\left(\sum_{p \in P} \ell_p \mathbf{o}_p(a_p)\right)(\theta) \\
&= \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\theta') = \theta}} \left(\sum_{p \in P} \ell_p \mathbf{o}_p(a_p)\right)(\theta') \\
&= \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\theta') = \theta}} \left(\sum_{p \in P} \ell_p \mathbf{o}_p(a_p)(\theta')\right) \\
&= \sum_{p \in P} \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\theta') = \theta}} \ell_p \mathbf{o}_p(a_p)(\theta') \\
&= \sum_{p \in P} \ell_p \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\theta') = \theta}} \sum_{\substack{\theta'' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \mathbf{o}_p(\theta'') = \theta'}} a_p(\theta'') \\
&= \sum_{p \in P} \ell_p \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\mathbf{o}_p(\theta')) = \theta}} a_p(\theta'') \\
&\underset{*}{=} \sum_{p \in P} \ell_p \sum_{\substack{\theta'' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \mathbf{o}_p(\sigma(\theta'')) = \theta}} a_p(\theta'') \\
&= \sum_{p \in P} \ell_p \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \mathbf{o}_p(\theta') = \theta}} \sum_{\substack{\theta'' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\theta'') = \theta'}} a_p(\theta'') \\
&= \sum_{p \in P} \ell_p \mathbf{o}_p \left(\sum_{\substack{\theta'' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\theta'') = \theta}} a_p(\theta'') \right) \\
&= \sum_{p \in P} \ell_p \mathbf{o}_p \left(\sigma(a_p)(\theta)\right)
\end{aligned}$$

$$= \sum_{p \in P} \ell_p \mathbf{o}_p \left(\sigma(a)_p(\theta)\right)$$
$$= \left(\mathbf{k} \odot \sigma(a)\right)(\theta)$$

Hence, we deduce $\sigma(\mathbf{k} \odot a) = \mathbf{k} \odot \sigma(a)$.

The main computability result is the following lemma: Given a vector of natural numbers, the set of induced unifying markings is finitely representable.

Lemma 87 (Representation of Realizations of Vectors of Natural Numbers)
Let $(F, \equiv)$ be compact. Let $\nu \in \mathbb{N}^P$. Let $V \subseteq \mathbb{V} \setminus P$. Let $\theta \in \langle {}^{V}_{F} \rangle$. Then, there exist a finite and computable set $S \subset \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P$ such that:

$$\mathcal{R}(S) = \mathcal{R}\left(\mathfrak{R}_\theta(\nu)\right)$$

Proof of Lemma 87. For $p \in P$ and $1 \leqslant i \leqslant \nu_p$ let $p_i \in \mathbb{V}_{\text{SORT}(p)} \setminus \mathbb{V}(\theta)$.

We define the unification problem U as follows:

$$U = \left\{(\mathbf{o}_p(p_i), \theta) \mid p \in P \text{ and } 1 \leqslant i \leqslant \nu_p\right\}$$

As $(F, \equiv)$ is compact, a computable finite set $L \subseteq \mathbb{V} \overset{F}{\rightsquigarrow} \mathbb{V}$ exists such that: $\mathcal{R}(L) = \mathbb{U}(U)$.

Let $a \in \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P$ for $p \in P$ by:

$$a_p = 1 \cdot p_1 + \cdots + 1 \cdot p_{\nu_p}$$

Then, we define the set $S \subseteq \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P$ of parameterized markings by:

$$S = \left\{\sigma(a) \mid \sigma \in L\right\}$$

In the following we show that $\mathcal{R}(S) = \mathcal{R}\left(\mathfrak{R}_\omega(\nu)\right)$.
"$\subseteq$": If $S = \emptyset$, then $\mathcal{R}(S) = \emptyset \subseteq \mathcal{R}\left(\mathfrak{R}_\theta(\nu)\right)$.

Let $m \in \mathcal{R}(S)$. Let $p \in P$. By definition, we have $\left\|m_p\right\| = \nu_p$. Let $\theta' \in \text{support}(m_p)$. We have to show that $\mathbf{o}_p(\theta') \equiv \theta$.

There exist $1 \leqslant i \leqslant \nu_p$, $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$, $\lambda \in L$ such that:

$$\theta' = \sigma(\lambda((p_i)))$$

As λ unifies U, we have that $\lambda(\mathbf{o}(p_i)) \equiv \lambda(\theta)$ and thus we have $\sigma(\lambda(\mathbf{o}(p_i))) \equiv \sigma(\lambda(\theta))$ by Corollary 85. Then, applying Lemma 86 yields:

$$\mathbf{o}_p(\sigma(\lambda(p_i)) = \sigma(\lambda(\mathbf{o}_p(p_i)))$$

Hence, we have $\mathbf{o}_p(\theta') \equiv \theta$ and deduce $m \in \mathcal{R}\left(\mathfrak{R}_\theta(\nu)\right)$.
"$\supseteq$": Let $m \in \mathcal{R}\left(\mathfrak{R}_\theta(\nu)\right)$. By definition, we have $\left\|m_p\right\| = \nu_p$. There exists $\zeta \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ such that $a' \in \mathfrak{R}_\theta(\nu)$ and $m = \zeta(a')$.

For the sufficiency, it remains show that there exists a $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ such that σ solves U and $\sigma(a) = m$.

We denote $\text{support}(m_p) = \{\theta_{p_1}, \ldots, \theta_{p_{\nu_p}}\}$ such that $1 \cdot \theta_{p_1} + \cdots + 1 \cdot \theta_{p_{\nu_p}} = m_p$. Then, we define $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ for p_i:

$$\sigma(x) = \begin{cases} \theta_{p_i} & \text{, if } \theta = p_i \\ \zeta(x) & \text{, otherwise.} \end{cases}$$

By definition of realization of abstract zero, it follows $\mathbf{o}_p(\theta_{p_i}) \equiv \zeta(\theta)$. Moreover, we have $\sigma(\mathbf{o}(p_i)) = \mathbf{o}_p(\sigma(p_i))$ by Lemma 86. Hence, $\sigma(\mathbf{o}(p_i)) \equiv \zeta(\theta) = \sigma(\theta)$ for all p_i, and σ solves U. For all $p \in P$, we have:

$$\begin{aligned} m_p &= 1 \cdot \theta_{p_1} + \cdots + 1 \cdot \theta_{p_{\nu_p}} \\ &= 1 \cdot \sigma(p_1) + \cdots + 1 \cdot \sigma(p_{\nu_p}) \\ &= \sigma(a_p) \end{aligned}$$

And hence $m = \sigma(a)$ and $\sigma(a) \in \mathcal{R}(S)$.

We conclude that $\mathcal{R}(S) = \mathfrak{R}_\omega(\nu)$.

It remains to show that S is finite and computable. We have $|S| \leqslant |L||P| \max\left\{\nu_p \mid p \in P\right\}$. As L is finite and computable, we deduce that S is finite and computable.

Computing Irreducible Solutions

In this section, we show that if $(F, \equiv)$ is compact a finite representation of the set of irreducible solutions is computable. First, we show that we can compute for a given polynomial r an equivalent polynomial r', such that every two terms of $\text{support}(r)$ are either equal or not equivalent.

Lemma 88 (Compute Non-Equivalent Support)
Let $(F, \equiv)$ be compact. Then, there exists a computable $r' \in \mathbb{Z}\langle {}^{\emptyset}_{F} \rangle$ with:

1. $r \equiv r'$, and
2. for all $\theta, \theta' \in \langle {}^{\emptyset}_{F} \rangle$, $\theta \neq \theta'$ implies $\theta \not\equiv \theta'$.

Proof of Lemma 88. First, we observe that $\equiv$ is decidable by computing a representation of a given term and testing syntactical unification. We sketch a computation procedure as follows: For all $\theta, \theta' \in \text{support}(r)$ with $\theta \neq \theta'$ do the following:

- If $\theta \equiv \theta'$, the value $r(\theta)$ is set to $r(\theta) + r(\theta')$. Afterwards, change the value of $r(\theta')$ to 0. And then continue with the changed r.

The correctness of this procedure follows from the following facts: For each pair of non- equivalent terms, the loop is executed. After the loop one of the terms is removed from the polynomial r. Hence, no two terms may be equal after executing all loops. This procedure halts and terminates as $\text{support}(r)$ is finite and does not increase in the loop. Moreover, the number of times the loop is executed is bounded by $|\text{support}(r)|^2$. Inside the loop is 3 operations for manipulating are needed. Moreover the operations needed to test equivalence, which are all computable. Hence, the sketched procedure terminates.

As a next step, we deduce in Lemma 89 that the set of irreducible solutions can be represented finitely, if $(F, \equiv)$ is compact.

Lemma 89 (Representation of Irreducible Solutions)
Let $r \neq 0$. Let $\Phi \in \{E, I\}$. Let $(F, \equiv)$ be compact.
Then, there exists a finite and computable set of parameterized markings $S \subset \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^{P}$ with $\mathcal{R}(S) = \mathbb{L}^{\mathbb{R}}(\Phi)$.

Proof of Lemma 89. First, we compute a variant of r, where each pair of terms the support is not equivalent. The computability follows from Lemma 88, which preserves the set of solutions as shown in Corollary 75. Let $\omega \in \text{support}(r)$, then we can compute the abstract solutions of D, the according linear Diophantine equation $\sum_{p \in P} \ell_p x_p \doteq r(\omega)$ or $\sum_{p \in P} \ell_p x_p \dot{\geqslant} r(\omega)$. As the size of them are bounded as shown in Theorem 63 (Page 132), by enumerating and testing to the bound, we may compute the set of

irreducible solutions of D. Then, each irreducible solution ν is an abstract solution and we may compute a finite representation S_ω of $\mathfrak{R}_\omega(\nu)$ by Lemma 87. By Lemma 80, $\mathfrak{R}_\omega(\nu)$ is the set of irreducible solutions with respect to $r(\omega) \cdot \omega$ as right member. By computing a finite representation S_ω for each monomial $r(\omega) \cdot \omega$ with $\omega \in \text{support}(r)$, we derive a finite set of representations. W.l.o.g we assume that the set of variables used disjoint and hence the point-wise sum of the sets $\sum_{\omega \in \text{support}(r)} S_\omega$ yields a finite and computable representation of the set of all irreducible solutions by Corollary 78.

Computing Irreducible Zeros

Symmetrically to Lemma 89, we can compute a finite representation of all irreducible zeros.

Lemma 90 (Representation of Irreducible Zeros)
Let $(F, \equiv)$ be compact.

Then, there exists a finite and computable set of parameterized markings $S \subset \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ with $\mathcal{R}(S) = \mathbb{L}_0^{\mathbb{IR}}(\Phi)$.

Proof of Lemma 90. Let D be the induced homogeneous linear Diophantine equation $\sum_{p \in P} \ell_p x_p = 0$ or homogeneous linear Diophantine inequality $\sum_{p \in P} \ell_p x_p \geqslant 0$, respectively. Here, x_p are the unknowns for $p \in P$. By Theorem 63 (Page 132), the size of all irreducible solutions is bounded. And hence, by enumerating and testing we can compute the set of irreducible solutions of D. Then, each delivers an abstract zero by Definition 81. Let ν be an abstract zero, $s \in \textsc{Sorts}(F)$, and $\nu_s \in \mathbb{V}_s$. Then, we have the following identity, as every term is a realization of a variable ν_s:

$$\bigcup_{s \in \textsc{Sorts}(F)} \mathcal{R}\left(\mathfrak{R}_{\nu_s}(\nu)\right) = \bigcup_{\theta \in \langle \frac{\emptyset}{F} \rangle} \mathfrak{R}_\theta(\nu)$$

As the number of sorts is finite, we may unify and compute a representation for each sort by Lemma 87. By Lemma 84 we know that the set of realizations are precisely the set of irreducible zeros.

9.4.1 Linear Representation of the Set of Solutions

Now, we prove the main result of this chapter. If $(F, \equiv)$ is compact, we can compute two finite sets of parameterized markings, such that

they form a linear representation (Definition 50, Page 113) of all solutions.

Theorem 91 (Computable Linear Representation of the Set of Solutions)
Let $(\mathsf{F}, \equiv)$ be compact.

Then, there exist computable finite sets $S, Z \subset \mathbb{N}\langle {}^{\mathbb{V}}_{\mathsf{F}} \rangle^{P}$ such that:

$$\textsc{Lin}(S, Z) = \mathbb{L}(\Phi)$$

Proof of Theorem 91. By Lemmas 89 and 90, we can compute sets $S, Z \subseteq \mathbb{N}\langle {}^{\mathbb{V}}_{\mathsf{F}} \rangle^{P}$ such that: $\mathcal{R}(S) = \mathbb{L}^{\mathbb{IR}}(\Phi)$ and $\mathcal{R}(S) = \mathbb{L}^{\mathbb{IR}}_{0}(\Phi)$. By Lemma 72, it follows $\textsc{Lin}(S, Z) = \mathbb{L}(\Phi)$.

10 NON-PRESERVING STEPS FROM A LINEAR REPRESENTATION

In this chapter, we show that stability of a given algebraic equation or inequality is decidable, if the underlying data scheme is compact. Moreover, we show that a witness —a non-preserving step— is computable. More specifically, we show that a non-preserving step that starts from a linear representation is computable. Then, we build on the result from Chapter 9, that for the set of solutions a linear representation is computable.

For this chapter, we fix the following notations:

- a distributed data scheme $(P, F, \equiv)$ (Definition 8, Page 41)
- an abstraction query $\mathbf{k} = (\boldsymbol{\ell}, \mathbf{o})$ (Definition 27, Page 66)
- a term polynomial $r \in \mathbb{Z}\langle \frac{\emptyset}{F} \rangle$
- the algebraic equation E: $\mathbf{k} \odot P \doteq r$ (Definition 29, Page 69)
- the algebraic inequality I: $\mathbf{k} \odot P \dot{\geqq} r$ (Definition 29, Page 69)
- $\Phi \in \{E, I\}$
- a simple transition $t \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P \times \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ (Definition 51, Page 121)
- for each $p \in P$: the term $\zeta_p^- \in \langle \frac{\emptyset}{F} \rangle$ with $\{\zeta^-\} \subseteq \text{support}(t_p^-)$

This chapter is structured into four sections. In Section 10.1, we show that it is sufficient to consider the *kernel* of a linear representation for computation of non-preserving steps. In Section 10.2, we reduce the computation of steps from parameterized markings to computing unifiers of a unification problem. In Section 10.3, we show how to compute a non-preserving step as a realization from a parameterized step. In Section 10.4, we combine all lemmas to prove the main result of this thesis: Computability of non-preserving steps.

10.1 THE KERNEL OF A LINEAR REPRESENTATION

In this section, we define the *kernel* of a linear representation and show in Lemma 94: For the computation of non-preserving steps, it is sufficient to consider the kernel.

The kernel is a subset of the set of solutions, which is defined dependent on t. We define τ as the maximum value of $t_p^-(\zeta_p^-)$ over all places $p \in P$ multiplied by $|P|$. The kernel is defined by considering at most τ zeros.

Definition 92 (Kernel)
Let $\tau = |P| \max\left\{t_p^-(\zeta_p^-) \mid p \in P\right\}$. Let $S, Z \subset \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^P$ be finite sets of parameterized markings. Let

$$K = \left\{s + z^1 + \cdots + z^n \mid s \in \mathcal{R}(S), z^1, \ldots, z^n \in \mathcal{R}(Z), \text{ and } n \leqslant \tau\right\}$$

Then, K is the *kernel* of $\textsc{Lin}(S, Z)$.

We observe the following corollary, which may be proven by renaming the variables and considering sums with at most τ representations of zeros.

Corollary 93 (Finite Representation of Kernel)
Let $S, Z \subset \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^P$ be finite sets of parameterized markings. Let K be the *kernel* of $\textsc{Lin}(S, Z)$.
Then, a finite representation of K is computable.

Now, we state the main lemma of this section: If there exists a non-Φ-preserving step, then there exists a non-Φ-preserving step starting from the kernel of the linear representation of the set of solutions.

Lemma 94 (Non-Preserving Steps from Kernel)
Let $S, Z \subset \mathbb{N}\langle {}^{\mathbb{V}}_{F}\rangle^P$ be finite such that $\textsc{Lin}(S, Z) = \mathbb{L}(\Phi)$. Let K be the kernel of $\textsc{Lin}(S, Z)$. Let $(m, m') \in \mathcal{R}(t)^{\uparrow}$ be a step of t with $m \in \mathbb{L}(\Phi)$ and $m' \notin \mathbb{L}(\Phi)$.
Then, there exists $(m^K, m'') \in \mathcal{R}(t)^{\uparrow}$ with $m^K \in \mathbb{L}(\Phi)$ and $m'' \notin \mathbb{L}(\Phi)$ and $m^K \in K$.

Due to its length, the proof of Lemma 94 is shown at the end of this section.

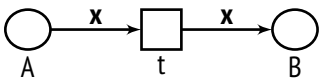
Figure 54 illustrates Definition 92 and Lemma 94. Figure 54a recalls the distributed data scheme $(P_{47}, F_{47}, \equiv_\emptyset)$, from Figure 47. The transition t is shown in Figure 54b: t consumes from place A and produces on place B. t is simple as all arc inscription are simple. As the incoming arc is $1 \cdot \mathbf{x}$, we have 1 as τ from Definition 92. Figure 54c shows the algebraic inequality I_{54}. As we will show in the following, I_{54} is not stable.

In Figure 54d, five markings of $(P_{54}, F_{54}, \equiv_\emptyset)$ are shown. $m^{54.1}$ is a solution, as the result of the abstraction query applied is $-2 \cdot f(c)$. In

places P_{54}: A, B

function symbols F_{54}: $f : S \to S, c :\to S$

(a) The distributed data scheme $(P_{54}, F_{54}, \equiv_\emptyset)$ for the examples.



(b) Algebraic Petri net structure S_{54}.

$$f(\mathbf{A}) - 2 \cdot \mathbf{B} \geqq_\emptyset -2 \cdot f(c)$$

(c) Algebraic inequality I_{54} of S_{54}.

	A	B	solution?	zero?	in kernel?
$m^{54.1}$	0	$1 \cdot f(c)$	✓	×	✓
$m^{54.2}$	$6 \cdot c$	$4 \cdot f(c)$	✓	✓	×
$m^{54.3}$	$5 \cdot c$	$3 \cdot f(c)$	×	×	×
$m^{54.4}$	$2 \cdot c$	$1 \cdot f(c)$	✓	✓	✓
$m^{54.5}$	$1 \cdot c$	$2 \cdot f(c)$	×	×	×

(d) Some markings of S_{54}.

Figure 54.: Example for a non-preserving step from the kernel.

the example, we consider the kernel with respect to the linear representation of irreducible solutions and irreducible zeros (cf. Chapter 9). The tuple of markings $(m^{54.2}, m^{54.3})$ is a step of t. Moreover, $m^{54.3}$ is not a solution, as the result of the abstraction query applied is $-3 \cdot f(c)$. Hence, $\left(m^{54.2}, m^{54.3}\right)$ is a non-preserving step and we deduce that I_{54} is not stable. However, $m^{54.2}$ is not in the kernel: $m^{54.2}$ is the sum of more than $\tau = 1$ irreducible zeros. By Lemma 94, there exists a non-preserving step, where the source marking is in the kernel. The tuple $\left(m^{54.4}, m^{54.5}\right)$ is also a step of t. Moreover, $m^{54.4}$ is the sum of one irreducible solution and one irreducible zero. Hence, $m^{54.4}$ is in the kernel and the step $\left(m^{54.4}, m^{54.5}\right)$ illustrates Lemma 94.

Now, we finish the section with the missing proof of Lemma 94.

Proof of Lemma 94. Let $\tau = |P| \max\left\{t_p^-(\zeta_p) \mid p \in P\right\}$. Let $s \in S$, $z^1, \ldots, z^n \in Z$ and $\overline{s} \in \mathcal{R}(s)$, $\overline{z}^1 \in \mathcal{R}\left(z^1\right)$, $\ldots$, $\overline{z}^n \in \mathcal{R}(z^n)$ such that:

$$\overline{s} + \overline{z}^1 + \cdots + \overline{z}^n = m$$

If $n \leqslant \tau$, then $m \in \mathcal{R}(K)$ and the assumption follows. Hence, we assume $n > \tau$ in the rest of the proof.

Let $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ and $\underline{m} \in \mathbb{N}\langle \substack{\emptyset \\ F} \rangle^P$ such that:

1. $m = \underline{m} + \sigma(t^-)$
2. $m' = \underline{m} + \sigma(t^+)$

Let $\{j_1, \ldots, j_\ell\} \subseteq \{1, \ldots, n\}$ be the least subset such that:

$$\overline{s} + \overline{z}^{j_1} + \cdots + \overline{z}^{j_\ell} \geqslant \sigma(t^-)$$

For every $p \in P$, the maximal number of zeros needed is bounded by $t_p^-(\zeta_p^-)$. Hence, the number of zeros is bounded by $|P| \max\left\{t_p^-(\zeta_p^-) \mid p \in P\right\}$. Hence, we deduce that $\ell < \tau$.

We now consider the step $\left(m^K, m''\right)$ defined by:

$$\begin{aligned} m^K &= \overline{s} + \overline{z}^{j_1} + \cdots + \overline{z}^{j_\ell} \\ m'' &= o + \sigma(t^\Delta) \end{aligned}$$

As $m^K \geqslant \sigma(t^-)$, we observe that $\left(m^K, m''\right)$ is a step of t. Furthermore, assuming Φ is an algebraic inequality, we have:

$$\begin{aligned} \mathbf{k} \odot m^K &= \mathbf{k} \odot \left(\overline{s} + \overline{z}^{j_1} + \cdots + \overline{z}^{j_\ell}\right) \\ &= \underbrace{\mathbf{k} \odot \overline{s}}_{\geqq r} + \underbrace{\mathbf{k} \odot \left(\overline{z}^{j_1} + \cdots + \overline{z}^{j_\ell}\right)}_{\geqq 0} \\ &\geqq r \end{aligned}$$

If Φ is an equation, we replace the three occurrences of $\geqq$ with $\equiv$. Hence, $m^K \in \mathbb{L}(\Phi)$ is a solution. Let $\left\{\hat{j}_1, \ldots, \hat{j}_{\hat{\ell}}\right\} = \{1, \ldots, n\} \setminus \{j_1, \ldots, j_\ell\}$. Then assuming Φ is an algebraic inequality, we have:

$$\begin{aligned} \mathbf{k} \odot m'' &= \mathbf{k} \odot \left(m^K + \sigma(t^\Delta)\right) \\ &= \mathbf{k} \odot \left(m - \left(\overline{z}^{\hat{j}_1} + \cdots + \overline{z}^{\hat{j}_{\hat{\ell}}}\right) + \sigma(t^\Delta)\right) \end{aligned}$$

$$
\begin{aligned}
&= \mathbf{k} \odot \left(m' - \left(\overline{z}^{\hat{j}_1} + \cdots + \overline{z}^{\hat{j}_\ell} \right) \right) \\
&= \mathbf{k} \odot (m') - \mathbf{k} \odot \left(\overline{z}^{\hat{j}_1} + \cdots + \overline{z}^{\hat{j}_\ell} \right) \\
&= \underbrace{\mathbf{k} \odot (m')}_{\not\geqq r} \underbrace{- \underbrace{\mathbf{k} \odot \left(\overline{z}^{\hat{j}_1} + \cdots + \overline{z}^{\hat{j}_\ell} \right)}_{\geqq 0}}_{\leqslant 0} \\
&\not\geqq r
\end{aligned}
$$

If Φ is an algebraic equation, we replace the two occurrences of $\not\geqq$ with $\neq$ and $\leqq$ and $\geqq$ by $=$. Hence, $o \notin \mathbb{L}(\Phi)$ is not a solution. And the step $\left(m^K, m'' \right)$ is non-preserving.

10.2 STEP UNIFICATION PROBLEM

In general, the transition t induces an infinite number of steps from the realizations of a parameterized marking. In this section, we define the *step unification problem* to characterize steps from the realizations of a parameterized marking.

The set of step unification problems characterizes the infinite set of steps finitely. In the literature the unifiers to the step unification problem are also known as *firing modes*, which may *enable* a transition, if a sufficient amount of terms is on each place [Rei13].

A step unification problem is a unification problem, that asks for a unifier such that t is in enabled in a realization of a given parameterized marking.

Definition 95 (Step Unification Problem)
Let $a \in \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^P$. For each $p \in P$, let $S_p \subseteq \text{support}(a_p)$ with $\sum_{\eta \in S_p} a_p(\eta) \geqslant t_p^-(\zeta_p^-)$. Let $U \subseteq \langle {}^{\mathbb{V}}_{F} \rangle \times \langle {}^{\mathbb{V}}_{F} \rangle$ with:

$$U = \left\{ \left(\eta, \zeta_p^- \right) \mid p \in P, \eta \in S_p \right\}$$

Then, U is a *step unification problem of* a. We denote the set of all step unification problems by StepUnification(a).

Now, we can the main property of the step unification problem in Lemma 96: Every unifier of the step unification problem induces a step starting from a realization of a given parameterized marking and —vice-versa— every step starting from a realization of the given

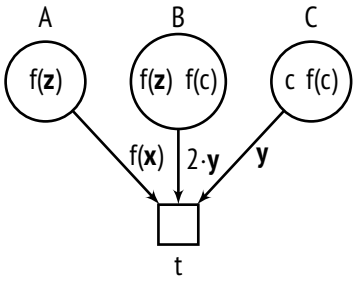


Figure 55.: Algebraic Petri net structure S_{55} with the $\{z\}$-parameterized marking a^{55}.

parameterized marking induces a unifier to a step unification problem.

Lemma 96 (Step Unification Problem and Steps)
Let $V \subseteq \mathbb{V} \setminus \mathbb{V}(t)$. Let $a \in \mathbb{N}\langle {V \atop F} \rangle^P$. Let $\sigma \in \mathbb{V}(a) \cup \mathbb{V}(t) \overset{F}{\rightsquigarrow} \emptyset$. Then, the following statements are equivalent:

1. $\left(\sigma(a), \sigma(a + t^{\Delta})\right) \in \mathcal{R}(t)^{\uparrow}$.
2. There exists a step unification problem U of a with $\sigma \in \mathbb{U}(U)$.

Proof of Lemma 96. 1.⇒2.: As $\left(\sigma(a), \sigma(a + t^{\Delta})\right) \in \mathcal{R}(t)^{\uparrow}$, we have that:

$$\sigma(a) \geqq \sigma(t^{-})$$

Hence, there exists $a' \leqslant a$ such that $\sigma(a') \equiv \sigma(t^{-})$. Let $p \in P$ and $S_p = \text{support}(a'_p)$. Moreover, we have $\sum_{\theta \in S_p} a'(\theta) = t^{-}_p(\zeta^{-}_p)$ and thus $\sum_{\theta \in S_p} a(\theta) \geqslant t^{-}_p(\zeta^{-}_p)$. Moreover, for each $\theta \in S_p$, we have $\sigma(\theta) \equiv \sigma(\zeta^{-}_p)$. Thus, σ solves the following step unification problem:

$$\left\{ \left(\eta, \zeta^{-}_p\right) \mid p \in P, \eta \in S_p \right\}$$

2.⇒1.: Let U be the unification such that for $p \in P$, $S_p \subseteq \text{support}(a_p)$ with $\sum_{\eta \in S_p} a_p(\eta) \geqslant t^{-}_p(\zeta^{-}_p)$. As σ solves U, we deduce: $\sigma(a) \geqq \sigma(t^{-})$. Moreover $\sigma(a) - \sigma(t^{-}) + \sigma(t^{+}) = \sigma(a + t^{\Delta})$ is a step of t.

unification problem					unifiable?
$\mathcal{U}_1$:	$(f(\mathbf{z}), f(\mathbf{x}))$	$(f(\mathbf{z}), \mathbf{y})$	$(f(c), \mathbf{y})$	$(c, \mathbf{y})$	$\times$
$\mathcal{U}_2$:	$(f(\mathbf{z}), f(\mathbf{x}))$	$(f(\mathbf{z}), \mathbf{y})$	$(f(c), \mathbf{y})$	$(f(c), \mathbf{y})$	✓

(a) Step unification problems $\mathcal{U}_1, \mathcal{U}_2 \in \textsc{StepUnification}(a^{55})$

	$\mathbf{x}$	$\mathbf{y}$	$\mathbf{z}$
σ_1	c	$f(c)$	c

(b) A unifier of $\mathcal{U}_2$.

	A	B	C
a^{56}	$f(\mathbf{z})$	$f(\mathbf{z}) + f(c)$	$c + f(c)$
$\sigma_1(t^{\Delta})$	$-f(c)$	$-2 \cdot f(c)$	$-f(c)$
$\sigma_1(a^{56})$	$f(c)$	$2 \cdot f(c)$	$c + f(c)$
$\sigma_1(a^{56} + t^{\Delta})$	0	0	c

(c) The step $\left(\sigma_1(a^{56}), \sigma_1(a^{56} + t^{\Delta})\right)$ induced by t.

Figure 56.: Example of step unification problem and an induced step of the transition and marking shown in Figure 55.

Figures 55 and 56 illustrate Definition 95 and Lemma 96. The underlying data scheme $(P_{54} \cup \{C\}, F_{54}, \equiv_{\emptyset})$ is the union of the data scheme from Figure 54a with the additional place C. Figure 55 shows the algebraic Petri net structure S_{55} with the transition t, which consumes from all three places, and the $\{\mathbf{z}\}$-parameterized marking a^{55}. Figure 56a shows two resulting step unification problems. Both, $\mathcal{U}_1$ and $\mathcal{U}_2$ ask to unify $f(\mathbf{z})$ with $f(\mathbf{x})$, $f(\mathbf{z})$ with $\mathbf{y}$, and $f(c)$ with $\mathbf{y}$. This reflects the two arcs going from the places A and B to t. As the arc from B to t is inscribed with $2 \cdot \mathbf{y}$, both terms of a^{56}_B have to be unified with $\mathbf{y}$. Regarding the arc from C to t, the two unification problems are different. $f(\mathbf{y})$ may either be unified with c in $\mathcal{U}_1$ or with $f(c)$ in $\mathcal{U}_2$, respectively. $\mathcal{U}_1$ is not unifiable as the last two pairs contradict each other. $\mathcal{U}_2$ is unifiable and Figure 56b shows the unifier σ_1. Then, by Lemma 96, the tuple $\left(\sigma_1(a), \sigma_1(a + t^{\Delta})\right)$ is a step of t. The step is shown in Figure 56c.

Applying Lemma 96 to non-preserving steps yields the following corollary.

Corollary 97 (Step Unification and Non-Preserving Steps)
Let $a \in \mathbb{N}\langle {}^{V}_{F} \rangle^{P}$ be a parameterized marking with $\mathcal{R}(a) \subseteq \mathbb{L}(\Phi)$. Then, the following statements are equivalent:

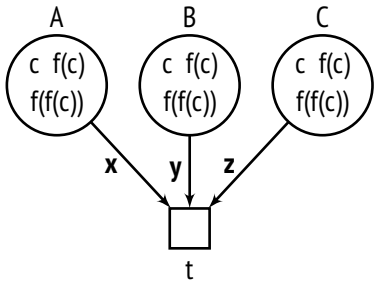
1. There exists a step $(m, m') \in \mathcal{R}(t)^{\uparrow}$ with $m \in \mathcal{R}(a)$ and $m' \notin \mathbb{L}(\Phi)$.
2. There exists a $U \in \textsc{StepUnification}(a)$ with a unifier $\sigma \in \mathbb{U}(U)$ such that $\sigma(a + t^{\Delta}) \notin \mathbb{L}(\Phi)$.

COMPUTABILITY OF STEP UNIFICATION PROBLEMS. In this paragraph, we show that the set of step unification problems is finite and computable.

Lemma 98 (Computability of the Set of Step Unification problems)
Let $a \in \mathbb{N}\langle {}^{V}_{F} \rangle^{P}$. Then, $\textsc{StepUnification}(a)$ is finite and computable.

Proof of Lemma 98. By Definition 95 each set S_p is a subset of $\text{support}(a_p)$. Hence, each subset S_p is finite and computable and there exist only finitely many different subsets. Moreover, there are at most $|P|$ terms used by the transition, which is again finite and computable. Again all combinations are finite, thus $\textsc{StepUnification}(a)$ is finite and computable.

Lemma 98 shows only computability and no upper or lower bound of the size. Indeed, an exponential blow-up is possible. We illustrate this by the example shown in Figure 57. The net structure S_{57} shown in Figure 57a consists of three places A, B, C, and one transition t and uses the data scheme $(F_{54}, \equiv_{\emptyset})$ shown in Figure 54a. Each arc is inscribed with a monomial over a different variable: $1 \cdot \mathbf{x}$, $1 \cdot \mathbf{y}$, and $1 \cdot \mathbf{z}$. The marking a^{57} assigns three different tokens to each place: c, f(c), and f(f(c)). The example yields $3^3 = 27$ different unification problems. If we added one more place with a new variable as arc inscription, the number of unification problems would increase to $3^4 = 81$. Moreover, the example shows that each unification problem yields a different unifier. Hence, the number of different induced steps is also exponential.



	A	B	C
t^-	**x**	**y**	**z**
a^{57}	[c, f(c), f(f(c))	[c, f(c), f(f(c))]	[c, f(c), f(f(c))]

(a) Algebraic Petri net structure S_{57} with marking a^{57} shown graphical and in a table.

U_1:	(**x**, c)	(**y**, c)	(**z**, c)
U_2:	(**x**, c)	(**y**, c)	(**z**, f(c))
U_3 :	(**x**, c)	(**y**, c)	(**z**, f(f(c)))
⋮	⋮	⋮	⋮

(b) Step unification problems

Figure 57.: Example for exponential growth of step unification problems.

10.3 NON-SATISFYING REALIZATIONS

In this section, we show that given a parameterized step, it is decidable, whether there exists a non-preserving realization. More precisely, we only consider steps, where it is known that each realization of the source marking is a solution.

First, we show that a given parameterized polynomial has a realization such that it is less than or incomparable to (with respect to $\equiv$) a given polynomial.

Lemma 99 (Non-Equivalent Polynomial Realization)
Let $(F, \equiv)$ be compact. Let $r' \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle$. Let $r \in \mathbb{Z}\langle {}^{\emptyset}_{F} \rangle$. Let $S = \left\{ \sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset \mid \sigma(r') \not\geq r \right\}$.
Then, emptiness of S is decidable.

Proof of Lemma 99. By Lemma 88 (Page 167), we assume that for all $\theta, \theta' \in \text{support}(r)$ we have that $\theta \neq \theta'$ implies $\theta \not\equiv \theta'$. We sketch a decision procedure:

> For each $\theta \in \text{support}(r)$ and each $R \subseteq \text{support}(r')$ do the following: Let $\overline{R} = \text{support}(R) \setminus R$. Let E be the dis-unification problem defined by
>
> $$E = \left\{ (\eta, \theta) \mid \eta \in \overline{R} \right\}$$
>
> Then, decide emptiness of $\mathcal{U}(E)$. If $\mathcal{U}(E) \neq \emptyset$ and $\sum_{\eta \in R} r' < r(\theta)$ stop the computation and output "S is not empty".
>
> If $\mathcal{U}(E) = \emptyset$ or $\sum_{\eta \in \overline{R}} r' < r(\theta)$ for all $\theta \in \text{support}(r)$ and $S \subseteq \text{support}(r')$, then output "S is empty".

The described procedure always terminates, as $\text{support}(r)$ and $\text{support}(r')$ are finite and thus the set of all subsets of $\text{support}(r')$ is finite. Moreover, emptiness of $\mathcal{U}(E)$ is decidable, as $(F, \equiv)$ is compact.

It remains to show that the sketched procedure is correct. If the procedure stops with "S is not empty", then there exist $S \subseteq \text{support}(r)$ and $\theta \in \text{support}(r)$ such that: $\mathcal{U}(E) = \emptyset$ or $\sum_{\eta \in \overline{S}} r' < r(\theta)$. Let $\sigma \in \mathcal{U}(E)$. Then, we have:

$$\sigma(r')(\theta) < \sum_{\eta \in \overline{S}} r' < r(\theta)$$

And hence, $\sigma(r') \not\geq r$. Thus, the procedure is correct, if the output is "S is not empty".

Now, it remains to show that if there exists a $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ such that $\sigma(r') \not\geq r$, then the output is "S is not empty". Let $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ such that $\sigma(r') \not\geq r$. Then, there exists $\theta \in \text{support}(r)$ such that:

$$\sum_{\substack{\theta' \in \text{support}(r') \\ \theta \equiv \theta'}} \sigma(r')(\theta) < r(\theta)$$

Then, let $R = \left\{ \theta' \in \text{support}(r') \mid \sigma(\theta') \equiv \theta \right\}$ and we have:

$$\sum_{\theta \in R} \sigma(r')(\theta) = \sum_{\substack{\theta' \in \text{support}(\sigma(r')) \\ \theta \equiv \theta'}} r'(\theta')$$

$$= \sum_{\substack{\theta' \in \text{support}(r') \\ \theta \equiv \theta'}} \sigma(r')(\theta')$$
$$< r(\theta)$$

Then, we have $\overline{R} = \text{support}(R) \setminus R$ and $E = \left\{(\eta, \theta) \mid \eta \in \overline{R}\right\}$. And by definition of R and $\overline{R}$, we have $\sigma(\eta) \not\equiv \theta$ for all $\eta \in \overline{R}$. Thus, σ is a dis-unifier of E. As the procedure checks all subsets R, the procedure stops with the output "R is not empty".

The proof of this lemma is the only time we use the prerequisite that dis-unification problems are decidable, which is implied by the compactness, (cf. Definition 5, Page 36). Using Lemma 99, we deduce in Lemma 100 that the statement also holds, if we replace $\geqq$ with $\equiv$.

Lemma 100 (Non-Equivalent Polynomial Realization)
Let $(F, \equiv)$ be compact. Let $r' \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle$. Let $r \in \mathbb{Z}\langle {}^{\emptyset}_{F} \rangle$.

$$\text{Let } S = \left\{\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset \mid \sigma(r') \not\equiv r\right\}$$

Then, emptiness of S is decidable.

Proof of Lemma 100. We sketch the procedure.

Let $S' = \left\{\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset \mid r'' \not\equiv r\right\}$. By Lemma 100, we can decide if S' is empty. If S' is not empty, output "S is not empty" and stop. Otherwise, if $\|r'\| = \|r\|$ output "S is empty". Otherwise, output "S is not empty".

The sketched procedure does always terminate by Lemma 100. It remains to show that the procedure is correct. We distinguish the following cases of the result of the procedure:

1st case: S' is not empty.
If S' is not empty, then there exists a $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$, such that $\sigma(r') \not\geqq r$, which implies $\sigma(r') \not\equiv r$. Thus, S is not empty and the output is correct.

2nd case: S' is empty and $\|r\| = \|r'\|$.
In this case for every $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ we have $\sigma(r) \geqq r$. As moreover, $\|r\| = \|r'\|$, we deduce $\|\sigma(r')\| = \|r\| = \|r'\|$,

which implies $\sigma(r) \equiv r$. Thus, the output "S is empty" is correct.

3rd case: S' is empty and $\|r\| \neq \|r'\|$.
In this case, for every $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$, we have $\sigma(r) \not\equiv r$. And thus, the output "S is not empty" is correct.

Now, we use the previous two lemmas to show the main results of this section: It is decidable whether a parameterized step may be realized as a non-preserving step from a satisfying marking if the underlying data scheme is compact. Moreover, a witness is computable if a non-preserving realization exists.

Lemma 101 (Computing Non-Preserving Realizations)
Let $V \subseteq \mathbb{V} \setminus P$. Let $(a, a') \in \mathbb{N}\langle \binom{V}{F} \rangle^P \times \mathbb{N}\langle \binom{V}{F} \rangle^P$ with $\mathcal{R}(a) \subseteq \mathbb{L}(\Phi)$.

Then, it is decidable, if there exists a $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ such that $\sigma(a') \notin \mathbb{L}(\Phi)$. Moreover, a non-preserving realization can be computed.

Proof of Lemma 101. If $\Phi = E$, let $\sim$ be $\equiv$, otherwise let $\sim$ be $\geqq$. It is sufficient to decide, whether a non-satisfying realization exists. To this end, we compute $\mathbf{k} \odot a'$ and decide if a $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ exists such that $\sigma(\mathbf{k} \odot a') \sim r$. By Lemma 86, we have $\sigma(\mathbf{k} \odot a') = \mathbf{k} \odot \sigma(a')$. By Lemma 99 if $\Phi = I$ or Lemma 100 if $\Phi = E$, we can decide if such a σ exists. If no such σ exists, we can stop the computation and output "false". Otherwise, we know such a σ exists. Then, we enumerate all $\langle \binom{\emptyset}{F} \rangle^{\mathbb{V}(a')}$ and check according assignments $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$. As the set of terms is defined inductively, it is enumerable. Moreover the finite Cartesian product is also enumerable. Thus, at some point this enumeration will stop with a σ such that $\mathbf{k} \odot \sigma(a') \not\sim r$. Thus, the computed σ is a computed witness.

10.4 COMPUTABILITY OF A NON-PRESERVING STEP

In this section, we combine the results of the previous sections to show decidability of the emptiness of the set of non-preserving steps. Furthermore, we show computability of a non-preserving step in case of non-emptiness. For all results in this section, we have the prerequisite that the underlying data scheme is compact.

As an intermediate step, we consider the restricted case of a given parameterized marking in the following lemma.

Lemma 102 (Computability of a Non-Preserving Step of a Parameterized Marking)
Let $(F, \equiv)$ be compact. Let $a \in \mathbb{N}\langle \frac{\mathbb{V}}{F} \rangle^P$ with $\mathcal{R}(a) \subseteq \mathbb{L}(\Phi)$. Then, let:

$$\mathfrak{N}_a = \left\{ (m, m') \in \mathcal{R}(t)^\uparrow \mid m \in \mathcal{R}(a) \text{ and } m' \notin \mathbb{L}(\Phi) \right\}$$

Then, emptiness of $\mathfrak{N}_a$ is decidable. In case of non-emptiness, a step $(m, m') \in \mathfrak{N}_a$ is computable.

Proof of Lemma 102. By Lemma 98, we can compute unification problems $U_1, \ldots, U_n$ with $\textsc{StepUnification}(a) = \{U_1, \ldots, U_n\}$.

As $(F, \equiv)$ is compact (Definition 5), we can compute for each $i \in \{1, \ldots, n\}$ a finite set $\Psi_i \subseteq \mathbb{V} \overset{F}{\rightsquigarrow} \mathbb{V}$ such that $\mathcal{R}(\Psi_i) = \mathbb{U}(U_i)$. Then, for each $\psi \in \bigcup_{i=1}^{n} \Psi_i$ we can compute $\psi(a + t^\Delta)$, and hence $\mathbf{k} \odot \psi(a + t^\Delta)$.

By Lemma 101, it is decidable whether there exists a $\sigma \in \mathbb{V} \overset{F}{\rightsquigarrow} \emptyset$ with $\sigma\left(\mathbf{k} \odot \psi(a + t^\Delta)\right) \notin \mathbb{L}(\Phi)$. If such a σ exists, an example is computable. Finally, Corollary 97 implies that such a σ exists if and only if $\left(\sigma(a), \sigma(a + t^\Delta)\right)$ is a non-preserving step.

Thus, if such a σ exists, it induces a non-preserving step. Otherwise, the set of non-preserving steps of a is empty.

Now, we show the main result of this thesis: The decidability of emptiness of the set of non-preserving steps and computability of a non-preserving step in case of non-emptiness. Or, put differently: We show decidability of preservation of all steps including unreachable. In a case of non-preservation, a witness in form of a non-preserving step can be computed.

For the proof, we use the previous lemma, the computability of a linear representation as shown in Chapter 9, and the kernel of the linear representation.

Theorem 103 (Computability of Non-Preserving Steps)
Let $(F, \equiv)$ be compact. Let $\mathfrak{N} \subseteq \mathcal{R}(t)^\uparrow$ be the set of non-preserving steps.

Then, emptiness of $\mathfrak{N}$ is decidable. In case of non-emptiness, a step $(m, m') \in \mathfrak{N}$ is computable.

Proof of Theorem 103. By Theorem 91, there exists computable finite sets $S, Z \subset \mathbb{N}\langle {}^{\mathbb{V}}_{\Gamma} \rangle^{P}$ such that $\textsc{Lin}(S, Z) = \mathbb{L}(\Phi)$. By Corollary 93 we can compute a finite representation of the kernel K. By Lemma 94, there exists a non-preserving step if and only if there exists a non-preserving step from K. As K is a finite representation, we may consider each $k \in K$ separately. Using Lemma 102, we can decide for each k separately, if there exists a non-preserving step and compute it. If the set of non-preserving is empty all steps preserve Φ. Otherwise, we have computed a non-preserving step.

Part III.

Closure

In part III, we discuss the results of the thesis. We present prototypical case studies in Chapter 11. We summarize the results, draw conclusions, and present ideas for future work in Chapter 12.

11 CASE STUDY: TWO PROTOTYPES

In this chapter, we present two prototypical programs that were designed to evaluate the computational approach presented in Part II.

In Section 11.1, we present the tool Hopsi. The tool was developed for the computation of a representation of solutions of a homogeneous inequality. We tested Hopsi with examples and observed that the run time increases dramatically with input size. In Section 11.2, we present the tool PreFIT. The tool was developed for the computation of non-preserving steps for a restricted class of colored Petri nets and inequalities for referential integrity. We tested PreFIT with synthetic random input and observed that the run time was comparably fast, even for bigger input nets.

11.1 PROTOTYPE 1: SOLUTIONS OF HOMOGENEOUS INEQUALITIES

In this section, we present Hopsi, a *tool for **Ho**mogeneous* P*-equation**S** and* P*-inequalitie**S***. Hopsi computes a linear representation of the solution space of homogeneous equations and inequalities. Here, the analysis is restricted to data schemes over a single sorted Herbrand structure. Hopsi is available on GitHub[1].

In Section 11.1.1, we sketch the main idea and architecture of Hopsi. In Section 11.1.2, we present and discuss the test results.

ACKNOWLEDGEMENT. The tool was designed, programmed and analyzed in the context of a student project by Arthur Bartels.

11.1.1 General Idea of Hopsi

Hopsi computes a representation of the solution space of a homogeneous algebraic equation or inequality. Computing the solution space is the crucial computational task for analyzing algebraic equations and inequalities as discussed in Chapter 6. Hopsi computes the set of solutions following the method proven to be correct in Chapter 9.

1 https://github.com/Unimarvin/hopsi

ARCHITECTURE. HOPSI was written in Java[2] and relies only on standard libraries.

To compute a representation of all solutions, the following steps are performed: After parsing the input file, HOPSI computes the upper bound as given by Theorem 63 (Page 132). Then, *all* vectors of natural numbers up to that bound are computed. Each vector is tested, if it is an abstract zero (Definition 81, Page 159). Then, for each abstract zero a most general unifier is computed. Here, all terms are considered where the respective coefficient is not zero. For the computation of most general unifiers, we implemented a variant of the Robinson algorithm for unification [Rob65]. Each abstract zero with its most general unifier yields a finite representation of all realizations of the abstract zero. Finally, the resulting set is a finite representation of the set of all solutions of the given homogeneous algebraic equation or inequality.

INPUT. As input format, the tool uses a variant of PNML[3]. As equations are not part of the PNML standard, we extended PNML to suit our needs. An additional XML-node is added to the root, which contains the specification of the algebraic equation or inequality.

OUTPUT. As output, HOPSI prints to the command line all abstract zeros as defined in Definition 81 (Page 159). Moreover, for each abstract zero, a most general unifier is given as output, if the set of unifiers is not empty. This way, each abstract zero with its most general unifier results in a parameterized marking. The union of all these markings forms a finite representation of the set of solutions of the given homogeneous algebraic equation or inequality.

11.1.2 Test Results

We tested HOPSI with fourteen algebraic equations and inequalities. The underlying Herbrand structure of all examples is shown in Figure 58a. The algebraic equations and inequalities are shown in the first column of Figure 58b. For testing HOPSI, we used a Lenovo Carbon X1 fourth generation with an intel core i5-6920oU processor with 2×2.30 GHz and 7.2GiB RAM with a Linux Mint 18 (64Bit)[4]. We ran each test three times.

2 https://www.java.com/
3 http://www.pnml.org/
4 https://linuxmint.com/

places P_{58}:	$\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$
function symbols F_{58}:	$\mathsf{f} : S \to S$,
	$\mathsf{g} : S \to S$,
	$\mathsf{c} :\to S$

(a) The distributed data scheme $(\mathsf{P}_{58}, \mathsf{F}_{58}, \equiv_\emptyset)$ for the examples of Hopsi.

algebraic equation or inequality		bound	number of abs. zeros	avg. run time
E_1:	$1 \cdot \mathbf{A} + 1 \cdot \mathbf{B} \doteq_\emptyset [\,]$	4	10	83ms
I_1:	$1 \cdot \mathbf{A} + 1 \cdot \mathbf{B} \dot{\geqq}_\emptyset [\,]$	4	10	86ms
E_2:	$1 \cdot \mathbf{A} - 1 \cdot \mathsf{f}(\mathsf{c}) \doteq_\emptyset [\,]$	4	10	83ms
I_2:	$1 \cdot \mathbf{A} - 1 \cdot \mathsf{f}(\mathsf{c}) \dot{\geqq}_\emptyset [\,]$	4	10	87ms
E_3:	$-1 \cdot \mathsf{f}(\mathbf{A}) + 1 \cdot \mathbf{B} + 1 \cdot \mathbf{C} \doteq_\emptyset [\,]$	6	56	50ms
I_3:	$-1 \cdot \mathsf{f}(\mathbf{A}) + 1 \cdot \mathbf{B} + 1 \cdot \mathbf{C} \dot{\geqq}_\emptyset [\,]$	6	56	48ms
E_4:	$1 \cdot \mathbf{A} - 1 \cdot \mathsf{f}(\mathbf{B}) - 1 \cdot \mathsf{f}(\mathsf{g}(\mathbf{C})) + 1 \cdot \mathbf{D} \doteq_\emptyset [\,]$	8	330	91ms
I_4:	$1 \cdot \mathbf{A} - 1 \cdot \mathsf{f}(\mathbf{B}) - 1 \cdot \mathsf{f}(\mathsf{g}(\mathbf{C})) + 1 \cdot \mathbf{D} \dot{\geqq}_\emptyset [\,]$	8	330	101ms
E_5:	$3 \cdot \mathsf{f}(\mathbf{A}) - 7 \cdot \mathbf{B} \doteq_\emptyset [\,]$	294	4469	6244ms
I_5:	$3 \cdot \mathsf{f}(\mathbf{A}) - 7 \cdot \mathbf{B} \dot{\geqq}_\emptyset [\,]$	294	2,988,637	26,586ms
E_6:	$5 \cdot \mathsf{f}(\mathbf{A}) - 9 \cdot \mathbf{B} \doteq_\emptyset [\,]$	486	8679	74,583ms
I_6:	$5 \cdot \mathsf{f}(\mathbf{A}) - 9 \cdot \mathbf{B} \dot{\geqq}_\emptyset [\,]$	486	12,362,416	n/a
E_7:	$4 \cdot \mathsf{f}(\mathbf{A}) + 3 \cdot \mathsf{g}(\mathbf{B}) - 5 \cdot \mathsf{f}(\mathsf{g}(\mathbf{C})) - 1 \cdot \mathbf{D} \doteq_\emptyset [\,]$	250	n/a	n/a
I_7:	$4 \cdot \mathsf{f}(\mathbf{A}) + 3 \cdot \mathsf{g}(\mathbf{B}) - 5 \cdot \mathsf{f}(\mathsf{g}(\mathbf{C})) - 1 \cdot \mathbf{D} \dot{\geqq}_\emptyset [\,]$	250	n/a	n/a

(b) Eight example inputs of Hopsi and the respective run time.

Figure 58.: Evaluation results with Hopsi.

The last three columns in Figure 58b show the test results of the computations with Hopsi: The computed bound is according to Theorem 63. The second to last column shows the number of abstract zeros that were computed up to the bound. The last column shows the average run time over three repeated executions. The execution of I_6 ran out of memory during the computation of the most general unifiers. During the computation with E_7 and I_7, Hopsi ran out of memory before computing all abstract zeros. Accordingly, the run time is not shown.

We observe that the run time increases as expected with the size of the equation and the size of the coefficients. The computed bound is an important factor which increases the computational effort. Here, the approach of Hopsi shows a substantial weakness: Enumerating all candidates for abstract zeros is not efficient. If several places are considered, the combinatoric explosion makes the computation intractable. For computing abstract zeros, Hopsi needs to consider $\binom{b+n-1}{n-1}$ vectors, where b is the bound as defined in Theorem 63 and n is the number of places. Accordingly, the number of places increases the run time tremendously. Moreover, Hopsi often computed more abstract zeros than necessary. For example, I_6 may also be represented using less than ten zeros.

An idea for future work is to embed techniques from integer linear programming to compute abstract zeros, which may improve the run times for larger algebraic equations and inequalities.

11.2 PROTOTYPE 2: REFERENTIAL INTEGRITY

In this section, we present the tool PreFIT[5]. PreFIT is a tool that computes non-preserving steps of colored Petri nets regarding a specification of referential integrity. The main purpose of the development of PreFIT was to create visibility and reproducibility of the research results presented in this thesis within reasonable run time. PreFIT accepts files from CPNTools [JK09; JK15], a tool for modeling and testing data-aware Petri nets.

We first sketch the general idea and architecture of PreFIT in Section 11.2.1. In Section 11.2.2, we study how an input net and an inequality for the specification of referential integrity can be created. In Section 11.2.3, we show how the output of PreFIT is presented. In Section 11.2.4, we briefly study the scalability of PreFIT using synthetic randomized input transitions and inequalities.

5 https://github.com/Unimarvin/prefit

11.2.1 General Idea of PreFIT

In contrast to Hopsi, PreFIT is a tool that does not compute the solution space, it computes non-preserving steps of a given algebraic Petri net. One objective of the tool is to avoid the computational complexity, which Hopsi ran into. In contrast to Hopsi, we only consider a restricted set of nets and a restricted set of inequalities; in particular, we expect all arc inscriptions and coefficients to have an absolute value of at most one. As a consequence, we avoid the combinatoric explosion Hopsi ran into. Moreover, the tool was designed such it can be ran in the browser resulting in a low-threshold for starting and trying the tool.

ARCHITECTURE. For the development of PreFIT, we relied on classical web technologies using JavaScript[6] and HTML[7] for the interface. All computations are done in the browser. Hence, no further installation or servers are needed. In the following, we briefly list the used technologies and libraries:

HTML AND JAVASCRIPT HTML is the standard technology used in the internet for displaying websites. JavaScript is widespread programming language, mainly used for developing web applications. PreFIT is compatible with any browser that supports HTML5 and ECMAScript2017. During development, google chrome version 57 for Linux was used for testing.

JQUERY jQuery[8] is a JavaScript library optimized for the manipulation of the HTML document and interaction with the user. PreFIT uses version 3.3.1.

BOOTSTRAP Bootstrap[9] is a suite of CSS[10] and JavaScript libraries, which provide the basic structure for the user interface. PreFIT uses version 3.3.7.

11.2.2 Input of PreFIT

The input is given as CPN-file created with CPNTools [JK09]. We used version 4.0.1 to create a model of the purchase order system studied in Chapter 5. In Figure 59 a screen shot of the editing window of CPNTools is visible. In CPNTools, it is not allowed to model

6 https://www.ecma-international.org/ecma-262/8.0/
7 https://www.w3.org/TR/2014/REC-html5-20141028/
8 https://jquery.com/
9 https://getbootstrap.com/
10 https://www.w3.org/Style/CSS/

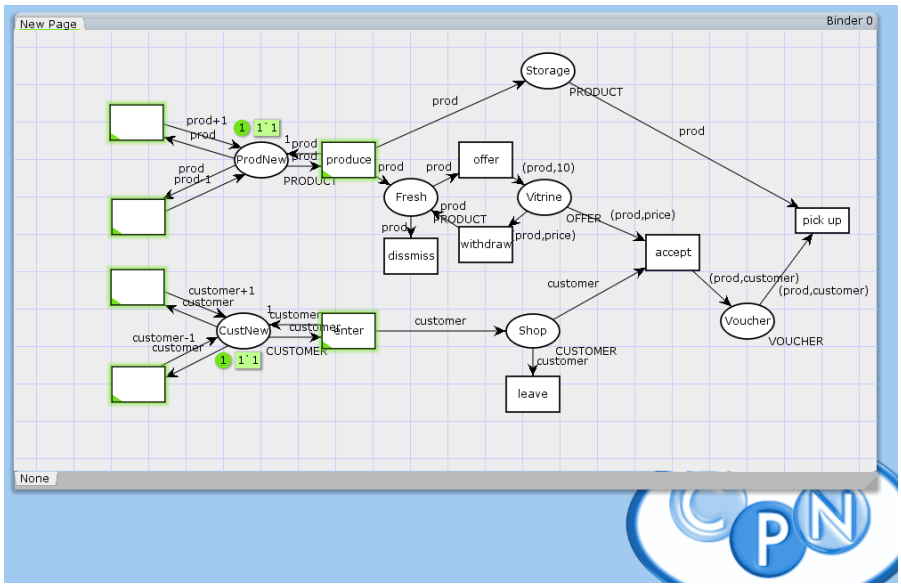


Figure 59.: Screen of CPNTools with a model of the Purchase Order System.

arc inscriptions using variables, which are only in outgoing arcs of transitions. Accordingly, we adapted the example of the purchase order system. First, we fixed the price to 10 for every order. Second, we introduced two new places for products and customer, which either increase or decrease the integer value modeling a product or customer. This way the value in that place may obtain an arbitrary integer value, which is then used by the transition "produce" or "enter", respectively.

PreFIT is optimized for a specific class of colored Petri nets. We describe the restrictions in the following.

INTEGER TUPLES WITHOUT ADDITION The sort specification may only use integers, aliases of integers and Cartesian products of integers. Using integers, we can encode identifiers and tuples of them. Furthermore, we can embed the data scheme of our illustrative example in Chapter 5. However, we assume that no operations, such as addition, are used as arc inscriptions.

SIMPLE TRANSITIONS PreFIT considers only simple transitions, as defined in Definition 51 (Page 121).

INEQUALITIES FOR REFERENTIAL INTEGRITY PreFIT computes non-preserving step for a restricted class of inequalities. We consider inequalities, where each integer coefficient is either 1, 0, or -1. This way it is possible to describe referential integrity as studied in Chapters 3 and 5.

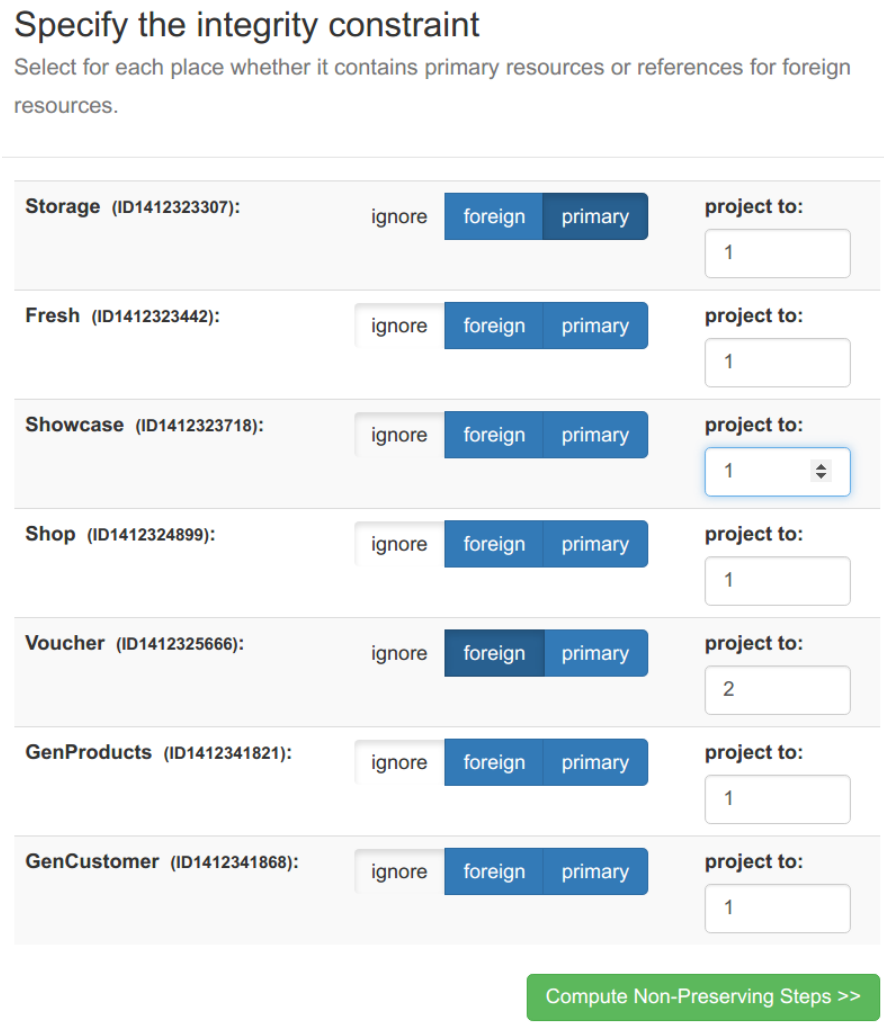


Figure 60.: Screen for creating an integrity constraint in PREFIT.

After the CPN-file is loaded, a dialog is shown to the user to specify the inequality. A screen shot for the purchase order example is shown in Figure 60. The user can select the coefficient for each place. The coefficient is either 0, if "ignore" is selected, -1 if "foreign" is selected and 1 if "primary" is selected. Moreover, the tuple stored at each place may be projected to the position described in each "project to" field. In Figure 60 the selection corresponds to the algebraic inequality $1 \cdot \mathbf{Storage} - 1 \cdot \mathrm{good}(\mathbf{Voucher}) \geqq 0$.

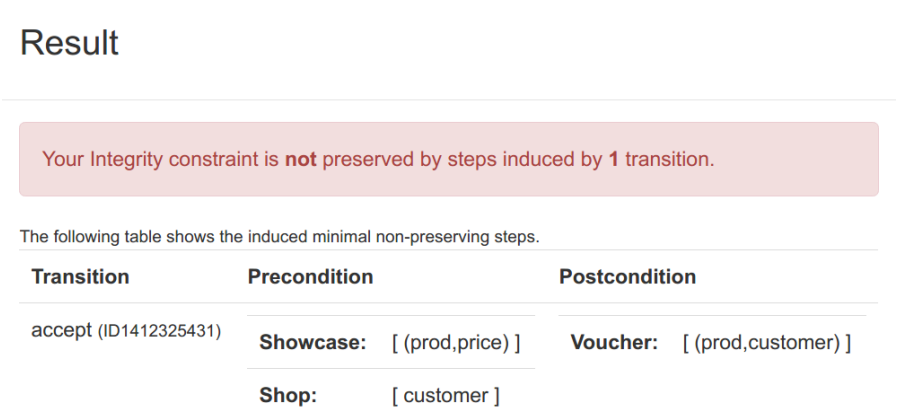


Figure 61.: Output screen of PreFIT.

11.2.3 Output of PreFIT

As output, the user is shown a step of each transition that induces a non-preserving step. A screen shot of the output for the purchase order example is shown in Figure 61. For each transition the marking before the step and after the step is shown in the columns "Precondition" and "Postcondition", respectively.

11.2.4 Scalability

In this section, we briefly study the run time scalability of PreFIT. We analyzed the run time with synthetic input which was generated with randomization. Here, the goal was to observe a tendency and to get an idea for the complexity, rather than having an in-depth analysis of the applicability with real-world examples.

We created two test scenarios. In the first scenario we ran the tool with a growing number of places. In the second scenario, we tested the tool with a high number of places, varying other parameters. In both scenarios, we restricted ourselves to the case of a single transition, as the computation of non-preserving step is achieved separately. As a test computer we used the same Lenovo Carbon X1 fourth generation with an intel core i5-69200U processor with 2×2.30 GHz and 7.2GiB RAM with a Linux Mint 18 (64Bit)[11] for comparability with the test of the first tool, Hopsi.

11 https://linuxmint.com/

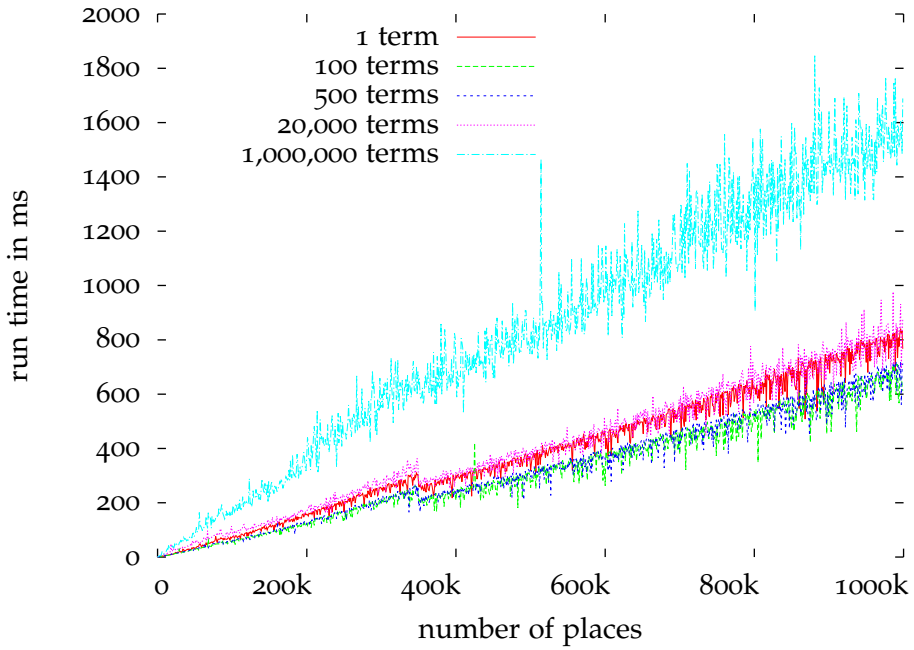


Figure 62.: Run time over number of places with synthetic, randomized input.

RUN TIME OVER NUMBER OF PLACES. In the test scenario, a random amount of the places were pre places of the transition. The remaining places were post places of the transition. Each arc was inscribed from a fixed set of terms. We chose the following sized sets of terms: 1, 100, 500, 20,000, and 1,000,000. Then, we randomized over the coefficient of each place. As zero coefficients correspond to ignoring the place, all places were assigned 1 or -1. Here, the ratio of different coefficients was also randomized avoiding distributions of 50% each coefficient for high numbers. Then, we ran each configuration five times and chose the maximum run time. We plotted the run time in milliseconds over the number of places as input. We ran each configuration ten times. As result value, we selected the maximum run time. Hence, outliers with higher run time are not hidden in the box plot. Accordingly, outliers with low run time are hidden. This decision was made to study a worst-case behavior. All run configurations needed less than a second. Figure 62 shows the four resulting graphs. Each graph represents a fixed set of terms from which the input was chosen. Overall, the graph suggests that the run time increases with a large number of places and the number of different terms. The number of terms however, also increases the run time but the effect is smaller. In Figure 62, it is noticeable that using 1 term is faster than using 100 or 500 different terms in most of the cases. The

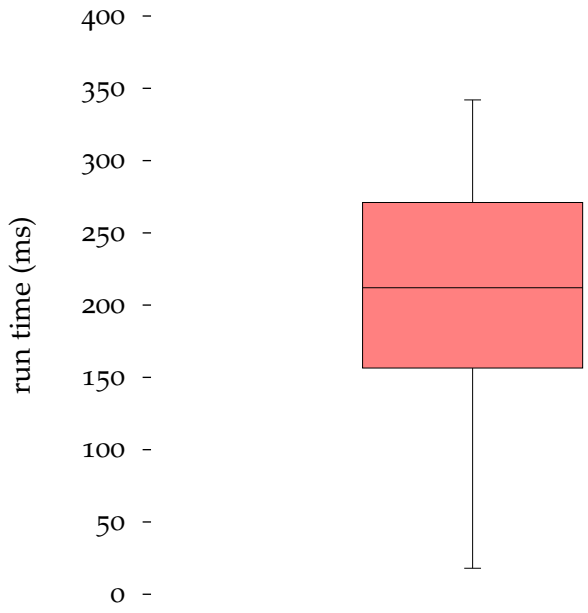


Figure 63.: Boxplot of run time using 1,000,000 places with at most 500 different terms all other parameters randomized with 1000 executions.

time difference is very small and with more terms it is more likely to find a counter example. For a larger number of terms, memory management by the infrastructure may also influence the run time here. Moreover, on all graphs a little decrease of run time when more than 35,000 places are used is visible. We conjecture this small difference is caused by memory management of the environment or a measuring error. As PREFIT is restricted to integer terms, unification problems are trivial. Overall, the number of terms have a small impact on the run time in the evaluation.

RUN TIME VARIATION WITH MANY PLACES. As a second step, we synthesized input with 1,000,000 places and up to 500 different terms, varying and randomizing other parameters. We repeated the execution a thousand times. The box plot in Figure 63 illustrates the distribution of the resulting run times. Here, 50% of the values measured are within the interval of the red box. The maximum run time was 342ms. The mean run time was 209ms with standard error of 2.3ms. The median was 212ms. There were no outliers. The results suggest that for an input size with up to 500.000 places and 500 different terms, PREFIT computes a non-preserving step in reasonable time for real-time applications.

DISCUSSION. The analysis above shows a rough indication that the approach may scale well. However, our approach using empirical analysis of randomly generated data cannot show the applicability of PREFIT to real-life examples. The limited evaluation provides no significant proof for the applicability. A detailed study of applicability within the context of software engineering is left for future work.

12 CONCLUSION

In this chapter, we conclude the thesis. We summarize the results in Section 12.1, discuss restrictions of Theorem 103 in Section 12.2, and suggest future work in Section 12.3

12.1 SUMMARY OF THE RESULTS

In this section, we briefly summarize the results of the thesis. We start with contributions regarding modeling in Section 12.1.1 and continue with contributions regarding analysis in Section 12.1.2.

12.1.1 Modeling

In Chapters 2 and 3, we studied *algebraic Petri nets* to model data-aware distributed systems and *algebraic equations and inequalities* to specify data integrity, respectively.

ALGEBRAIC PETRI NETS. To model data, we relied on data schemes, adapted from *quotient algebras* [ST12]. We defined the subclass of *compact data schemes*, where the set of solutions of each unification problem is finitely representable and the solvability of each dis-unification problem is decidable. We did not assume uniqueness of data entities, and rather modeled distributed data using bags distributed over places, resulting in *markings*.

We have shown in Theorem 16 that the steps of a system can be modeled by transitions of an algebraic Petri net if and only if the set of steps is monotone, precondition bounded, effect bounded, and structured.

The four characterizations fit well into properties of distributed computational systems. Monotonicity intuitively means that adding more resources to the precondition of a step can not avoid the step. The boundedness of the conditions of each step corresponds to limited physical resource of each computational systems. Moreover, data manipulations described by term functions which are not finitely representable are beyond the intuition of computational systems.

ALGEBRAIC EQUATIONS AND INEQUALITIES. Algebraic equations and inequalities are an established tool for the specification of correctness in distributed systems modeled by algebraic Petri nets [Vau85; Rei13]. In Chapter 3, we have shown examples that specify *referential integrity* in distributed systems. Moreover, properties like *mutual exclusion* [Rei13] can be specified.

12.1.2 Analysis

Regarding the analysis, we contributed in two points both assuming a compact data scheme: general undecidability of validity and computability of a non-preserving step.

UNDECIDABILITY. In Theorem 44, we have shown that validity of any algebraic equation or inequality is undecidable, even when considering a simple *Herbrand structure* [ST12] as data scheme. The result stands in line with related work, where most properties are undecidable when a combination of data modeling and process modeling is considered [Las16; Min67; Bag+13; RF10].

STABILITY. The main result of this thesis is Theorem 103: Stability is decidable and a non-preserving step is computable, if the underlying data scheme is compact. The proof is given in Part II. The most important ingredient for the proof is Theorem 91: A linear representation of the set of solutions is computable.

The theorem has two prerequisites: First, the data scheme is compact. Second, we consider stability instead of validity. As stability over general data schemes is undecidable and validity is undecidable for compact data schemes, both prerequisites are necessary for the result. We will discuss them in Section 12.2.

12.2 THE RESTRICTIONS OF THEOREM 103

For the proof of Theorem 103, we have used two restrictions: First, we considered stability (Definition 45, Page 90) instead of validity. Second, we restricted ourselves to the class of compact data schemes (Definition 5, Page 36). We discuss the restrictions of compactness in Section 12.2.1 and the difference between stability in validity in Section 12.2.2.

12.2.1 Compactness

A data scheme is compact (Definition 5, Page 36), if a finite representation of all unifiers is computable for each unification problem and for each dis-unification problem it is decidable whether it is dis-unifiable.

For data schemes, that are not compact, it is well-known that unification and dis-unification problems are undecidable [BS94]. This implies the following: Given a distributed data scheme, two markings m, m' and a transition t, it is undecidable, whether (m, m') is a step of t. As we can encode the unification problem into the decision problem, if realizing the parameterized markings t^- and t^+ into m and m', respectively.

12.2.2 Stability vs Validity

A major contribution of this thesis is the proof that stability of algebraic equations and inequalities is decidable, whereas validity is not. For stable algebraic equations and inequalities, validity is decidable. This arises the question of the difference between stability and validity.

As validity is undecidable, it is not possible to find a complete verification technique. This is a problem that affects formal methods in general [Ric53].

Stability may be used to show validity, but if the algebraic equation or inequality is unstable, we cannot tell whether it is valid. Hence, there is a set of algebraic equations and inequalities, where stability is useful and there is a set of algebraic equations and inequalities, where stability is not useful and verification using stability is inconclusive.

However, by computing a non-preserving step as a witness for instability, this non-preserving step may be used for further analysis and design of the system: On the one hand, if an algebraic equation or inequality is valid, but not stable, the computed non-preserving step is not reachable. In that scenario, it may be sufficient for the analysis to prove that the step is not reachable. Again, reachability is an undecidable problem. On the other hand, if the algebraic equation or inequality is invalid, the non-preserving step shows a possibility, where the specification can be violated.

We have studied an example in Chapter 5 to show applicability. It is left for future work to study the applicability of stability for the analysis of distributed data-aware systems for software engineering.

12.3 FUTURE WORK

In this section, we sketch ideas for future work.

COMPOSITIONALITY. For the design and analysis of distributed systems, it is a common design pattern to distribute a computational system over interacting *services*. A well-known architectural approach is known as *service-oriented computing* [Pap08; Bou+17].

Each service is designed separately. During design time, knowledge is only about one service available. It is an open question how a model of one service may help to deduce data integrity of the whole system. One approach would be to consider compositions of open algebraic Petri nets following the approaches in [Wag15; BG14; ST13; LMW07].

DEDUCTION, IMPLICATION. In classical databases, an integrity theory consists of a set of integrity constraints with deduction rules [DMM15; AHV95].

Given a set of valid algebraic equations and inequalities, it implies validity of other algebraic equation and inequalities. It is an open question, how implication may be decided. However, given a linear representation (Definition 50, Page 113) of the set of solutions could be a starting point.

TRANSITION GUARDS. In our model of algebraic Petri nets, we do not consider transition guards. As discussed in Section 2.3.2, transition guards are a helpful modeling technique which equips a transition with conditions. The steps induced by the transition are the steps satisfying the condition. In other words, guards describe a subset of $\mathcal{R}(t)^{\uparrow}$ where t is a transition. Hence, guards may only restrict the behavior and thus validity (stability) in a model without guards implies validity (stability) in the same model with guards. However, it is left for future work how models with guards affects our results for analysis and verification.

COMPLEXITY ANALYSIS. Theorem 103 shows a computability result. From a complexity theory point-of-view it is not clear, how difficult the problem is depending on the input parameters. Assuming a Herbrand structure as data scheme, we delineate a proof idea for coNP-completeness:

Every irreducible solution and zero is polynomial in size. If there exists a non-preserving step, then there exists a non-preserving step starting from the kernel. The kernel is build from a polynomial

amount of zeros and a solution. Thus, a counter-example is polynomial in size and by non-deterministic guessing and verifying, stability can be decided. This proof sketch would imply that stability is in co-NP.

On the other hand, assuming a data scheme inducing a singleton term, the subset sum problem may be reduced to solvability of an algebraic equation. Then again, solvability can be reduced to stability. This would imply co-NP hardness.

It is an open question, whether this proof sketch can be fully formalized.

EVENTUAL CONSISTENCY. In this thesis, we defined data integrity as a state property. Sometimes, it is not necessary that every reachable marking satisfies data integrity. Rather, it may be sufficient that a satisfying marking is reachable from every reachable marking. This variant is known as *eventual consistency* [DMM15]. It is an open problem to study eventual consistency with respect to algebraic equations and inequalities.

MORE EXPRESSIVE ABSTRACTION QUERIES. In Definition 27, we allowed one term-induced function and one integer coefficient per place. In general, one could consider more expressive abstraction query. We suggest ideas for generalization in the following list:

- allowing multiple term induced functions and coefficients per place,
- adding the possibility to express "filter" such that only partial contents of each place are considered,
- allowing products of places by using the Cauchy product; this way, it would be possible to express statements which correspond to the Cartesian product of sets

It is well-known that considering bag semantics, query problems are often undecidable [Fro17]. Accordingly, for some classes of abstraction queries, verification results may be undecidable. Intuitively, the reason is as follows. If operations where multiplication of unknowns can be expressed, it is possible to encode Hilbert's tenth problem: solvability of a non-linear Diophantine equation, which is known to be undecidable [Mat05].

Appendices

A MATHEMATICAL BASICS

In this section, we briefly define the mathematical notions and notations we use in this thesis without giving further explanations or examples.

A.1 BASICS

SETS, FUNCTIONS, NUMBERS. We assume the definition of a *set* as given and use the common notations: $\in, \subset, \subseteq \{,\}$. For a set M, we denote the subset of M containing all elements that satisfy a condition c by $\{x \in M \mid \text{condition } c\}$.

We denote the *set of natural numbers* 0,1,2,... as $\mathbb{N}$. We denote the *set of integers* 0,1,−1,2,−2,... as $\mathbb{Z}$.

We denote the *magnitude of a set* M by $|M| \in \mathbb{N} \cup \{\infty\}$, denoting the number of elements if finite, and ∞ otherwise.

Let M, M' be sets. We assume the definition of a *function from* M *to* M' as given. For application of a function we use *prefix notation*. If stated we may use *infix*, *postfix*, *subscript* or *superscript*. By abuse of notation we denote constant functions with their constant. For sets M, N, we write M^N for the set of all unary functions from N to M, called the N-function space over M. If N is finite, we also refer to the elements of M^N as N-*vectors* over M. For $\mathbb{N}$ and $\mathbb{Z}$, we use $+$ and $-$ with their common interpretation as binary functions and denote multiplication by concatenation.

Let M, M' be sets. Let $f_1, \ldots, f_n$ be functions from M' to M. Let g be a function M^n to M. Then, the *point-wise application of* g *to* $f_1, \ldots, f_n$ is a function g' from M' to M defined for $m \in M'$ by: $g'(m) = g(f_1(m), \ldots, f_n(m))$. By abuse of notation, we write g instead of g', if clear from context.

RELATIONS. Let $M_1, \ldots, M_n$ be sets with $n \in \mathbb{N}$ and $m_i \in M_i$ for all $1 \leqslant i \leqslant n$. Then, $(m_1, \ldots, m_n)$ is a *tuple over* $M_1, \ldots, M_n$. We denote the set of all *tuples over* $M_1, \ldots, M_n$ by $M_1 \times \cdots \times M_n$. Each subset $R \subseteq M_1 \times \cdots \times M_n$ is a *relation*. For a binary relation R and $(x, y) \in R$, we also use infix notation xRy.

Let M be a set and $R \subseteq M \times M$ be a relation. Then, R is *reflexive* if $(m, m) \in R$ for all $m \in M$; R is *symmetric* if $(m, m') \in$

R implies $(m', m) \in R$ for all $m, m' \in M$; R is *anti-symmetric* if $(m, m'), (m', m) \in R$ implies $m = m'$ for all $m, m' \in M$; R is *transitive* if $(m, m'), (m', m'') \in R$ implies $(m, m'') \in R$ for all $m, m', m'' \in M$. If R is reflexive, anti-symmetric, and transitive, R is a *partial order*. If R is reflexive, symmetric, and transitive, R is an *equivalence relation*. The *reflexive-transitive closure of* R is the least reflexive and transitive relation containing R, denoted by R^*.

For $\mathbb{N}$ and $\mathbb{Z}$, we use $<,>,\leqslant,\geqslant$ with their common interpretation as orders.

A.2 STRUCTURES

Let $\mathcal{M}$ be a set of sets. Let F be a set of functions such that each $f \in F$ is a function from $M_1 \times \cdots \times M_n$ to M_{n+1} with $M_1, \ldots, M_{n+1} \in \mathcal{M}$. Then, we call $\mathcal{M}$ with F a *structure*. If clear from context we identify the set $\mathcal{M}$ with the structure.

ABELIAN GROUP. Let M be a set, f be a function from $M \times M$ to M and $e \in M$ a constant.

- f is *associative on* M if for all $k, l, m \in M$ holds that $f(k, f(l, m)) = f(f(k, l), m)$.
- e is *left-identity of* f *in* M if $f(e, m) = m$ for all $m \in M$, and *right-identity of* f if $f(m, e) = m$ for all $m \in M$. e is an *identity of* f *in* M if e is a left- and right-identity of f.
- f is *commutative in* M if $f(l, m) = f(l, m)$ for all $l, m \in M$.
- For an element $m \in M$ and identity e, we call $m' \in M$ the *inverse of* m if $f(m, m') = e$.

We call the structure M with f and e an Abelian group if f is associative and commutative on M, e is an identity of f, and for every $m \in M$ exists an inverse of m.

ENDOMORPHISM. Let $M_1, \ldots, M_{n+1}$ be sets. Let f be a function from $M_1 \times \cdots \times M_n$ to M_{n+1} with $M_i \in M$. Let $\mathfrak{M}$ be a set with $M_i \subseteq \mathfrak{M}$ for all $1 \leqslant i \leqslant n+1$ and h be a function from $\mathfrak{M}$ to $\mathfrak{M}$. Then, h is an f-*endomorphism* if the following two statements are true:

1. $m \in M_i$ implies $h(m) \in M_i$ for all $1 \leqslant i \leqslant n+1$.
2. For all $(m_1, \ldots, m_n) \in M_1 \times \cdots \times M_n$ holds that $h(f(m_1, \ldots, m_n)) = f(h(m_1), \ldots, h(m_n))$.

$\mathbb{Z}$-MODULE. Let M be a set, f be a binary function and $e \in M$ such that they form an Abelian group. Then, f induces an integer multiplication on M as follows for $\lambda \in \mathbb{Z}$ and $m \in M$ with m' as the inverse of m:

$$\lambda m = \begin{cases} e & \text{, if } \lambda = 0 \\ f(m, (\lambda - 1)m) & \text{, if } \lambda > 0 \\ (-\lambda)m' & \text{, if } \lambda < 0 \end{cases}$$

We observe that the multiplication is an f-endomorphism.

CONGRUENCE. Let $M_1, \ldots, M_{n+1} \subseteq M$ be sets. Let $R_i \subseteq M_i \times M_i$ be an equivalence relation on M_i for all $i \in \{1, \ldots, n+1\}$. Let $R = \bigcup_{1 \leqslant i \leqslant n+1} R_i$ be the union of the equivalence relations. Let f be a function from $M_1 \times \cdots \times M_n$ to $M_n + 1$. Let for all $\{m_1, m_1'\} \subseteq R_1, \ldots, \{m_n, m_n'\} \subseteq R_n$ hold:

$$\big(f(m_1, \ldots, m_n), f(m_1', \ldots, m_n')\big) \in R_{n+1} \,.$$

Then R is a *f-congruence*. We extend this notion for a set of functions F, writing F-congruence, if R is a f-congruence for all $f \in F$.

A.3 POLYNOMIALS, BAGS

Let M be a set. Let f be a function from M to $\mathbb{Z}$. We denote with $\text{support}(f) = \{m \in M \mid f(m) \neq 0\}$ the *support of* f. If $|\text{support}(f)| \in \mathbb{N}$, then f is an *M-polynomial*. We denote the *set of all M-polynomials* by $\mathbb{Z}M$. Let $f, f' \in \mathbb{Z}M$ be two M-polynomials. Then, $f + f'$ is defined point-wise by $(f + f')(m) = f(m) + f(m)$ for all $m \in M$, accordingly $(-f)(m) = -f(m)$. We observe that $\mathbb{Z}M$ with $+$ and 0 is an Abelian group. If $\text{support}(f) = \{m\}$, we call f a *monomial*, denoted by $f(m) \cdot m$. If $f(m) = 1$, we may write m instead of $1 \cdot m$ by abuse of notation.

Let f with $f(m) \geqslant 0$ for all $m \in M$. Then, f is a *bag*. We denote bags by denoting the multiplicities in braces, i.e. $f = [a, a, b]$ if $f(a) = 2$, $f(b) = 1$ for some $a, b \in M$ and $f(m) = 0$ for all $m \in M \setminus \{a, b\}$. We denote the *empty bag* 0 by $[\,]$. We denote the *set of all bags over* M with $\mathbb{N}M$.

Let $\equiv \subseteq M \times M$ be an equivalence relation on M. Then, M induces an equivalence relation of $\mathbb{Z}M$ as follows for $f, f' \in \mathbb{Z}M$: $f \equiv f'$ if and only if for all $m \in M$: $\sum_{m' \in M; m' \equiv m} f(m') = \sum_{m' \in M; m' \equiv m} f'(m')$. Accordingly, $\equiv$ induces a partial order $\trianglerighteq$ as follows: $f \trianglerighteq f'$ if and only if there exists a $f'' \in M\mathbb{N}$ with $f \equiv f' + f''$.

A.4 TERMS

SORT SYMBOLS, FUNCTION SYMBOLS, VARIABLE SYMBOLS. We denote $\mathbb{S}$ as the *set of all sort symbols*, $\mathbb{V}$ as the set of *all variables*, and $\mathbb{F}$ as the *set of all function symbols*. We assume pairwise disjointness: $\mathbb{S} \cap \mathbb{V} = \mathbb{S} \cap \mathbb{F} = \mathbb{F} \cap \mathbb{V} = \emptyset$. Every variable $v \in \mathbb{V}$ has a sort $\textsc{Sort}(v) \in \mathbb{S}$. Each function symbol $f \in \mathbb{F}$ is of the form $f : s_1 \times \cdots \times s_n \to s_{n+1}$, where $s_1, \ldots, s_{n+1} \in \mathbb{S}$. We call n the *arity*, f the *name*, and $s_1 \times \cdots \times s_n \to s_{n+1}$ the *signature* of $f : s_1 \times \cdots \times s_n \to s_{n+1}$. The *sorts of* f are $\textsc{Sorts}(f) = \{s_1, \ldots, s_{n+1}\}$. If a function symbol $f \in \mathbb{F}$ has arity 0, we call f a *constant*.

TERMS. Let $F \subseteq \mathbb{F}$, $V \subseteq \mathbb{V}$. The *sorts of* F are defined by $\textsc{Sorts}(F) = \bigcup_{f \in F} \textsc{Sorts}(f)$. Let $s \in \textsc{Sorts}(F)$. The *set of* V*-parameterized* F*-terms of sort* s, $\langle {}^{V}_{F} \rangle|_s$, is defined recursively as follows:

- Let $v \in V$ and $\textsc{Sort}(v) = s$. Then, $v \in \langle {}^{V}_{F} \rangle|_s$.
- Let $f : s_1 \times \cdots \times s_n \to s \in F$, and $\theta_i \in \langle {}^{V}_{F} \rangle|_{s_i}$ for $1 \leqslant i \leqslant n$. Then, $f(\theta_1, \ldots, \theta_n) \in \langle {}^{V}_{F} \rangle|_s$.

For a set $S \subseteq \textsc{Sorts}(F)$, we write $\langle {}^{V}_{F} \rangle|_S = \bigcup_{s \in S} \langle {}^{V}_{F} \rangle|_s$ for the union of terms. If $S = \textsc{Sorts}(F)$, we also write $\langle {}^{V}_{F} \rangle$, instead of $\langle {}^{V}_{F} \rangle|_S$. For a term $\theta \in \langle {}^{\mathbb{V}}_{F} \rangle$, we write $\mathbb{V}(\theta)$ for the smallest set $\mathbb{V}(\theta) \subseteq \mathbb{V}$ satisfying $\theta \in \langle {}^{\mathbb{V}(\theta)}_{F} \rangle$. We lift this notion to a set of terms $\Theta \subseteq \langle {}^{\mathbb{V}}_{F} \rangle$ by union: $\mathbb{V}(\Theta) = \bigcup_{\theta \in \Theta} \mathbb{V}(\theta)$.

HERBRAND STRUCTURE. Let $F \subseteq \mathbb{F}$. Let $f \in F$ with $\textsc{Signature}(f) = s_1 \times \cdots \times s_n \to s_{n+1}$. Then, f induces a function on terms $\hat{f} : \langle {}^{V}_{F} \rangle|_{s_1} \times \cdots \times \langle {}^{\mathbb{V}}_{F} \rangle|_{s_n} \to \langle {}^{\mathbb{V}}_{F} \rangle|_{s_{n+1}}$ defined for $\theta_1 \in \langle {}^{\mathbb{V}}_{F} \rangle|_{s_1}, \ldots, \theta_n \in \langle {}^{\mathbb{V}}_{F} \rangle|_{s_n}$ by:

$$\hat{f}(\theta_1, \ldots, \theta_n) = f(\theta_1, \ldots, \theta_n)$$

By abuse of notation, we write f instead of $\hat{f}$, if clear from context. We observe that there exists a $m \in \mathbb{N}$ such that for any $V \subseteq \mathbb{V}$: (1) $\{s_1, \ldots, s_m\} = \textsc{Sorts}(F)$ and $\langle {}^{V}_{F} \rangle|_{s_1}, \ldots, \langle {}^{V}_{F} \rangle|_{s_m}$ with the functions induced by F build a structure. By abuse of notation we refer to this structure by $\langle {}^{V}_{F} \rangle$. We call $\langle {}^{\emptyset}_{F} \rangle$ the *Herbrand structure of* F.

SUBSTITUTIONS. Let $F \subseteq \mathbb{F}$. Let $\phi : \langle {}^{\mathbb{V}}_{F} \rangle \to \langle {}^{\mathbb{V}}_{F} \rangle$ be an endomorphism (w.r.t. to all functions induced by F) with $\phi(v) = v$ for almost all $v \in \mathbb{V}$. Then, ϕ is an F-substitution from V to V'.

Let $V, V' \subseteq \mathbb{V}$. Let ϕ be a F-substitution with $\phi(v) = v$ for all $v \in \mathbb{V} \setminus V$ and $\phi(v) \in \langle {}^{V'}_{F} \rangle$ for all $v \in V$. Then, ϕ is a F-*substitution from* V *to* V'. We denote the set of all F-substitutions from V to V' by $V \overset{F}{\rightsquigarrow} V'$.

TERM CONGRUENCES. Let $F \subseteq \mathbb{F}$ and $\langle {}^{\emptyset}_{F} \rangle$ its Herbrand structure. Let $\equiv \subseteq \langle {}^{\emptyset}_{F} \rangle \times \langle {}^{\emptyset}_{F} \rangle$ be a F-congruence. We implicitly extend $\equiv$ to $\langle {}^{\mathbb{V}}_{F} \rangle$ as the smallest F-congruence on $\langle {}^{\mathbb{V}}_{F} \rangle$ containing $\equiv$.

Let $\equiv_1, \equiv_1 \subseteq \langle {}^{\emptyset}_{F} \rangle \times \langle {}^{\emptyset}_{F} \rangle$ be F-congruences. If $\equiv_1 \subseteq \equiv_2$, $\equiv_2$ is a *specialization* of $\equiv_1$.

TERM POLYNOMIALS. Let $F \subseteq \mathbb{F}$. Let $V \subseteq \mathbb{V}$. Let $r \in \mathbb{Z}\langle {}^{V}_{F} \rangle$. Then r is a *V-parameterized* F*-polynomial*. The sorts of r are defined as the sorts of the support: $\textsc{Sorts}(r) = \textsc{Sorts}(\text{support}(r))$. We write $\mathbb{V}(r)$ short for $\mathbb{V}(\text{support}(r))$. We lift the absolute value $|\cdot|$ to polynomials by point-wise application. We lift a substitution $\sigma \in V'' \overset{F}{\rightsquigarrow} V'''$ as follows:

$$\sigma(r)(\theta) = \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \sigma(\theta') = \theta}} r(\theta') .$$

For a term congruence $\equiv \subseteq \langle {}^{\emptyset}_{F} \rangle \times \langle {}^{\emptyset}_{F} \rangle$ we lift $\equiv$ as follows for $r, r' \in \mathbb{Z}\langle {}^{\mathbb{V}}_{F} \rangle$: $r \equiv r'$ if and only if for all $\theta \in \langle {}^{\mathbb{V}}_{F} \rangle$:

$$\sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \theta' \equiv \theta}} r(\theta') = \sum_{\substack{\theta' \in \langle {}^{\mathbb{V}}_{F} \rangle \\ \theta' \equiv \theta}} r'(\theta')$$

Accordingly, we write $r \geqq r'$ if there exists a term bag $b \in \mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle$ with: $r \equiv r' + b$

A.5 EFFECTS, MARKINGS

Let $F \subseteq \mathbb{F}$. Let $P \subseteq \mathbb{V}$ be finite. Let $V \subseteq \mathbb{V}$. Let $e \in \mathbb{Z}\langle {}^{V}_{F} \rangle^{P}$ with: $\theta \in \text{support}(e_p)$ implies $\textsc{Sort}(\theta) = \textsc{Sort}(p)$ for all $p \in P$. Then, e is a *V-parameterized effect of* P *and* F.

Let $m \in \mathbb{Z}\langle {}^{V}_{F} \rangle^{P}$ be a V-parameterized effect over F and P with $m \geqslant 0$. Then, we call m a *V-parameterized marking over* F *and* P. We denote the set of all *V-parameterized markings over* F *and* P as $\mathbb{N}\langle {}^{\mathbb{V}}_{F} \rangle^{P}$.

We omit "$\emptyset$-parameterized" and simply write effect (marking) over F and P.

We lift addition and order of term polynomials point-wise to effects. We lift substitutions and absolute value by point-wise application. For a term congruence $\equiv \subseteq \langle {}^{\emptyset}_{F} \rangle \times \langle {}^{\emptyset}_{F} \rangle$, we extend $\equiv$ and $\geqq$ point-wise to effects.

A.6 REALIZATION AND REPRESENTATION

REALIZATION. Let $F \subseteq \mathbb{F}$, $\equiv \subseteq \langle^{\emptyset}_{F}\rangle \times \langle^{\emptyset}_{F}\rangle$ and $P \subseteq \mathbb{V}$. Let $X \in \{\langle^{\mathbb{V}}_{F}\rangle, \mathbb{Z}\langle^{\mathbb{V}}_{F}\rangle, \mathbb{Z}\langle^{\mathbb{V}}_{F}\rangle^{P}\}$. Let $x \in X$. Then, *the set of all* F-$\equiv$*-realizations of* x, denoted by $\mathcal{R}(x)$ is defined by:

$$\mathcal{R}(x) = \left\{ y \in X \mid \text{There exists } \sigma \in \mathbb{V}(x) \overset{F}{\rightsquigarrow} \emptyset \text{ with } \sigma(x) \equiv y \right\}.$$

We lift this notion for a set $X' \subseteq X$ by union: $\mathcal{R}(X') = \bigcup_{x \in X'} \mathcal{R}(x)$.

REPRESENTATION. Let $F \subseteq \mathbb{F}$, $A \subseteq \langle^{\emptyset}_{F}\rangle \times \langle^{\emptyset}_{F}\rangle$ and $P \subseteq \mathbb{V}$. Let $X \in \{\langle^{\mathbb{V}}_{F}\rangle, \mathbb{Z}\langle^{\mathbb{V}}_{F}\rangle, \mathbb{Z}\langle^{\mathbb{V}}_{F}\rangle^{P}\}$. Let $M, M' \subseteq X$.

Then, M is a M'-representation, if $\mathcal{R}(M) = \mathcal{R}(M')$.

AXIOMS AS CONGRUENCE REPRESENTATION. Let $F \subseteq \mathbb{F}$. Let $s \in \textsc{Sorts}(F)$. Let $\theta, \theta' \in \langle^{\mathbb{V}}_{F}\rangle|_{s}$. Then, the pair (θ, θ') is an *axiom*. By abuse of notation, we denote the axiom by $\theta \doteq \theta'$. For a set of axioms A, let $\hat{A} \subseteq \langle^{\mathbb{V}}_{F}\rangle \times \langle^{\mathbb{V}}_{F}\rangle$ be the smallest term congruence such that $\theta \doteq \theta' \in \hat{A}, (\rho, \rho') \in \mathcal{R}((\theta, \theta'))$ implies $(\rho, \rho') \in \hat{A}$. Then, $\hat{A}$ is the F-*congruence induced by* A, denoted by $\equiv_{A}$.

A.7 STEP, MONOTONICITY, TRANSITION

Let $F \subseteq \mathbb{F}$. Let $P \subseteq \mathbb{V}$ be finite. Let $m, m \in \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P}$. Then, (m, m') is a *step*. Let $S \subseteq \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P} \times \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P}$ be a set of steps with: $m \in \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P}, (m', m'') \in S$ implies $(m' + m, m'' + m) \in S$. Then, S is *monotone*.

Let $S' \subseteq \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P} \times \mathbb{N}\langle^{\emptyset}_{F}\rangle^{P}$ be a set of steps. And $S'' \supseteq S$ be the least monotone set containing S'. Then, S'' is the *monotone closure* of S', denoted by $S'^{\uparrow}$.

Let $a, a' \in \mathbb{N}\langle^{\mathbb{V}}_{F}\rangle^{P}$. Then, $t = (a, a')$ is a *transition*. We refer to a as t^{-} and to a' as t^{+}.

Let $(m, m') \in \mathcal{R}(t)^{\uparrow}$. Then, (m, m') is a step of t.

A.8 COMPUTABILITY

We assume the reader is familiar with the notions of *(un-)decidability* and *computability*. We will use natural language to describe procedures which show the above. We refrain from introducing formal concepts such as Turing machines.

B INDEX

C BIBLIOGRAPHY

[Abd+96] Parosh Aziz Abdulla et al. "General Decidability Theorems for Infinite-State Systems." In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 1996, pp. 313–321 (cit. on p. 96).

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995 (cit. on pp. 21, 24, 55, 61, 63, 75, 202).

[AII09] Ali A. Alwan, Hamidah Ibrahim, and Nur Izura Udzir. "Improved Integrity Constraints Checking in Distributed Databases by Exploiting Local Checking." In: *Journal of Computer Science and Technology* 24.4 (July 2009), pp. 665–674 (cit. on p. 76).

[AIU09] Ali A. Alwan, Hamidah Ibrahim, and Nur Izura Udzir. "Ranking and Selecting Integrity Tests in a Distributed Database." In: *Proceedings of the 11th International Conference on Information Integration and Web-based Applications &Amp; Services*. iiWAS '09. Kuala Lumpur, Malaysia: ACM, 2009, pp. 185–192 (cit. on p. 76).

[Alf09] Jorge L. Ramírez Alfonsín. *The Diophantine Frobenius Problem*. Oxford Univ. Press, 2009 (cit. on p. 140).

[Ali+18] Mahdi Alizadeh et al. "Linking data and process perspectives for conformance analysis." In: *Computers & Security* 73 (2018), pp. 172–193 (cit. on p. 58).

[Alv+16] Peter Alvaro et al. "Automating Failure Testing Research at Internet Scale." In: *Proceedings of the Seventh ACM Symposium on Cloud Computing*. SoCC '16. Santa Clara, CA, USA: ACM, 2016, pp. 17–28 (cit. on p. 21).

[And15] Sandro Andreotti. "Linear Programming and Integer Linear Programming in Bioinformatics." PhD thesis. Free University of Berlin, 2015 (cit. on p. 139).

[AWW02] Karen Aardal, Robert Weismantel, and Laurence A. Wolsey. "Non-standard approaches to integer programming." In: *Discrete Applied Mathematics* 123.1 (2002), pp. 5–74 (cit. on p. 140).

[Bag+13] Babak Bagheri Hariri et al. "Verification of Relational Data-centric Dynamic Systems with External Services." In: *Proceedings of the 32Nd Symposium on Principles of Database Systems*. PODS '13. New York, New York, USA: ACM, 2013, pp. 163–174 (cit. on pp. 23, 58, 95, 98, 200).

[BBM16] Franz Baader, Stefan Borgwardt, and Barbara Morawska. "Extending Unification in EL to Disunification: The Case of Dismatching and Local Disunification." In: *Logical Methods in Computer Science* 12.4 (2016) (cit. on pp. 24, 25, 30, 40, 55).

[BCM87] Eugenio Battiston, Fiorella de Cindio, and Giancarlo Mauri. "OBJSA Nets: a Class of High- level Nets Having Objects as Domains." In: *Advances in Petri Nets 1988, covers the 8th European Workshop on Applications and Theory of Petri Nets, held in Zaragoza, Spain in June 1987, selected papers*. Ed. by Grzegorz Rozenberg. Vol. 340. Lecture Notes in Computer Science. Springer, 1987, pp. 20–43 (cit. on p. 56).

[BG14] Franziska Bathelt-Tok and Sabine Glesner. "Towards the Automated Synthesis of Data Dependent Service Controllers." In: *PhD Symposium of the 11th International Conference on Service Oriented Computing (ICSOC 2013)*. Springer-Verlag Berlin Heidelberg, 2014, pp. 528–534 (cit. on pp. 58, 202).

[BHL92] Imre Bárány, Roger Howe, and László Lovász. "On integer points in polyhedra: a lower bound." In: *Combinatorica* 12.2 (1992), pp. 135–142 (cit. on p. 140).

[Bie+16] Felix Bieker et al. "A Process for Data Protection Impact Assessment Under the European General Data Protection Regulation." In: *Privacy Technologies and Policy - 4th Annual Privacy Forum, APF 2016, Frankfurt/Main, Germany, September 7-8, 2016, Proceedings*. Ed. by Stefan Schiffner et al. Vol. 9857. Lecture Notes in Computer Science. Springer, 2016, pp. 21–37 (cit. on p. 21).

[BM04] Michel Bidoit and Peter D. Mosses. *Casl User Manual - Introduction to Using the Common Algebraic Specification Language*. Vol. 2900. Lecture Notes in Computer Science. Springer, 2004 (cit. on p. 54).

[Bou+17] Athman Bouguettaya et al. "A Service Computing Manifesto: The Next 10 Years." In: *Commun. ACM* 60.4 (Mar. 2017), pp. 64–72 (cit. on p. 202).

[BS92] Franz Baader and Klaus U. Schulz. "Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures." In: *Automated Deduction - CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings*. Ed. by Deepak Kapur. Vol. 607. Lecture Notes in Computer Science. Springer, 1992, pp. 50–65 (cit. on pp. 30, 34, 55).

[BS94] Franz Baader and Jörg H. Siekmann. "Unification theory." In: *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume2, Deduction Methodologies*. Ed. by Dov M. Gabbay et al. Oxford University Press, 1994, pp. 41–126 (cit. on pp. 24, 25, 30, 34, 36, 40, 55, 201).

[BS96] Franz Baader and Klaus U. Schulz. "Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures." In: *J. Symb. Comput.* 21.2 (1996), pp. 211–243 (cit. on p. 55).

[Cal+04] Andrea Calí et al. "Data integration under integrity constraints." In: *Inf. Syst.* 29.2 (2004), pp. 147–163 (cit. on p. 76).

[Cat10] Rick Cattell. "Scalable SQL and NoSQL data stores." In: *SIGMOD Record* 39.4 (2010), pp. 12–27 (cit. on p. 55).

[CCL13] Andrea Calí, Diego Calvanese, and Maurizio Lenzerini. "Data Integration under Integrity Constraints." In: *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*. Ed. by Janis A. Bubenko Jr. et al. Springer, 2013, pp. 335–352 (cit. on p. 76).

[CM77] Ashok K. Chandra and Philip M. Merlin. "Optimal Implementation of Conjunctive Queries in Relational Data Bases." In: *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*. Ed. by John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison. ACM, 1977, pp. 77–90 (cit. on p. 54).

[Cod70] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks." In: *Commun. ACM* 13.6 (1970), pp. 377–387 (cit. on p. 55).

[Com91] Hubert Comon. "Disunification: A Survey." In: *Computational Logic - Essays in Honor of Alan Robinson*. Ed. by Jean-Louis Lassez and Gordon D. Plotkin. The MIT Press, 1991, pp. 322–359 (cit. on p. 55).

[Coo+92] William J. Cook et al. "On integer points in polyhedra." In: *Combinatorica* 12.1 (1992), pp. 27–37 (cit. on p. 140).

[Cuz+13] Alfredo Cuzzocrea et al. "Effectively and efficiently supporting crowd-enabled databases via NoSQL paradigms." In: *3RD International Workshop on Semantic Search over the Web, SSW '13, Riva del Garda, Italy, August 30, 2013*. Ed. by Roberto De Virgilio et al. ACM, 2013, 7:1–7:5 (cit. on p. 55).

[CV93] Surajit Chaudhuri and Moshe Y. Vardi. "Optimization of *Real* Conjunctive Queries." In: *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC, USA*. Ed. by Catriel Beeri. ACM Press, 1993, pp. 59–70 (cit. on p. 54).

[Dan+97] E. Dantsin et al. "Complexity and expressive power of logic programming." In: *Computational Complexity, 1997. Proceedings., Twelfth Annual IEEE Conference on (Formerly: Structure in Complexity Theory Conference)*. June 1997, pp. 82–101 (cit. on pp. 58, 98).

[Dan63] George B. Dantzig. *Linear programming and extensions*. Rand Corporation Research Study. Princeton, NJ: Princeton Univ. Press, 1963. XVI, 625 (cit. on p. 139).

[De +17] Riccardo De Masellis et al. "Add Data into Business Process Verification: Bridging the Gap between Theory and Practice." In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. Ed. by Satinder P. Singh and Shaul Markovitch. AAAI Press, 2017, pp. 1091–1099 (cit. on pp. 95, 96).

[DHV13] Elio Damaggio, Richard Hull, and Roman Vaculín. "On the equivalence of incremental and fixpoint semantics for business artifacts with Guard-Stage-Milestone lifecycles." In: *Inf. Syst.* 38.4 (2013), pp. 561–584 (cit. on p. 58).

[Dij80] Edsger W. Dijkstra. "Some Beautiful Arguments Using Mathematical Induction." In: *Acta Inf.* 13 (1980), pp. 1–8 (cit. on pp. 89, 96).

[DMM15] Hendrik Decker, Francesc D. Muñoz-Escoí, and Sanjay Misra. "Data Consistency: Toward a Terminological Clarification." In: *Computational Science and Its Applications - ICCSA 2015 - 15th International Conference, Banff, AB, Canada, June 22-25, 2015, Proceedings, Part V*. Ed. by Osvaldo Gervasi et al. Vol. 9159. Lecture Notes in Computer Science. Springer, 2015, pp. 206–220 (cit. on pp. 24, 73–75, 202, 203).

[DMW82] Walter Dosch, Gianfranco Mascari, and Martin Wirsing. "On the Algebraic Specification of Databases." In: *Eigth International Conference on Very Large Data Bases, September 8-10, 1982, Mexico City, Mexico, Proceedings*. Morgan Kaufmann, 1982, pp. 370–385 (cit. on p. 55).

[Dum+13] Marlon Dumas et al. "Essential Process Modeling." In: *Fundamentals of Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 63–96 (cit. on p. 99).

[EH08] Javier Esparza and Keijo Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2008 (cit. on p. 57).

[EH16] Javier Esparza and Philipp Hoffmann. "Reduction Rules for Colored Workflow Nets." In: *Fundamental Approaches to Software Engineering - 19th International Conference, FASE 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*. Ed. by Perdita Stevens and Andrzej Wasowski. Vol. 9633. Lecture Notes in Computer Science. Springer, 2016, pp. 342–358 (cit. on p. 57).

[Erb+14] Serdar Erbatur et al. "On Asymmetric Unification and the Combination Problem in Disjoint Theories." In: *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*. Ed. by Anca Muscholl. Vol. 8412. Lecture Notes in Computer Science. Springer, 2014, pp. 274–288 (cit. on p. 55).

[Fah+11] Dirk Fahland et al. "Behavioral Conformance of Artifact-Centric Process Models." In: *Business Information Systems - 14th International Conference, BIS 2011, Poznan, Poland, June 15-17, 2011. Proceedings*. 2011, pp. 37–49 (cit. on p. 58).

[Fin87] Alain Finkel. "A Generalization of the Procedure of Karp and Miller to Well Structured Transition Systems." In: *Automata, Languages and Programming, 14th International Colloquium, ICALP87, Karlsruhe, Germany, July 13-17, 1987, Proceedings*. Ed. by Thomas Ottmann. Vol. 267. Lecture Notes in Computer Science. Springer, 1987, pp. 499–508 (cit. on p. 57).

[Flo67] Robert W. Floyd. "Assigning Meanings to Programs." In: *Proceedings of Symposium on Applied Mathematics* 19 (1967), pp. 19–32 (cit. on pp. 89, 96).

[FMV14] Carlo A. Furia, Bertrand Meyer, and Sergey Velder. "Loop invariants: Analysis, classification, and examples." In: *ACM Comput. Surv.* 46.3 (2014), 34:1–34:51 (cit. on p. 96).

[FR74] Michael J. Fischer and Michael O. Rabin. "Super-Exponential Complexity of Presburger Arithmetic." In: *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Ed. by Bob F. Caviness and Jeremy R. Johnson. Vienna: Springer Vienna, 74, pp. 122–135 (cit. on p. 140).

[Fro17] André Frochaux. "Static Analysis of Monadic Datalog on Finite Labeled Trees." PhD thesis. Humboldt-Universität zu Berlin, 2017 (cit. on pp. 54, 203).

[FS01] Alain Finkel and Philippe Schnoebelen. "Well-structured transition systems everywhere!" In: *Theor. Comput. Sci.* 256.1-2 (2001), pp. 63–92 (cit. on pp. 57, 96).

[GHM76] John V. Guttag, Ellis Horowitz, and David R. Musser. "The Design of Data Type Specifications." In: *Proceedings of the 2nd International Conference on Software Engineering, San Francisco, California, USA, October 13-15, 1976.* Ed. by Raymond T. Yeh and C. V. Ramamoorthy. IEEE Computer Society, 1976, pp. 414–420 (cit. on p. 53).

[GL02] Seth Gilbert and Nancy Lynch. "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services." In: *SIGACT News* 33.2 (June 2002), pp. 51–59 (cit. on pp. 21, 75).

[GL79] Hartmann J. Genrich and Kurt Lautenbach. "The Analysis of Distributed Systems by Means of Predicate / Transition-Nets." In: *Semantics of Concurrent Computation, Proceedings of the International Symposium, Evian, France, July 2-4, 1979*. Ed. by Gilles Kahn. Vol. 70. Lecture Notes in Computer Science. Springer, 1979, pp. 123–147 (cit. on pp. 76, 97).

[GLT79] Hartmann J. Genrich, Kurt Lautenbach, and P. S. Thiagarajan. "Elements of General Net Theory." In: *Net Theory and Applications, Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, October 8-19, 1979*. Ed. by Wilfried Brauer. Vol. 84. Lecture Notes in Computer Science. Springer, 1979, pp. 21–163 (cit. on pp. 76, 97).

[GMW12] Christian Gierds, Arjan J. Mooij, and Karsten Wolf. "Reducing Adapter Synthesis to Controller Synthesis." In: *IEEE T. Services Computing* 5.1 (2012), pp. 72–85 (cit. on p. 97).

[Gom02] Ralph E. Gomory. "Early Integer Programming." In: *Operations Research* 50.1 (2002), pp. 78–81 (cit. on p. 139).

[GRB04] Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin. "Expand, Enlarge, and Check: New Algorithms for the Coverability Problem of WSTS." In: *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference, Chennai, India, December 16-18, 2004, Proceedings*. Ed. by Kamal Lodaya and Meena Mahajan. Vol. 3328. Lecture Notes in Computer Science. Springer, 2004, pp. 287–298 (cit. on p. 96).

[Gri81] David Gries. *The Science of Programming*. Texts and Monographs in Computer Science. Springer, 1981 (cit. on p. 96).

[GS66] Seymour Ginsburg and Edwin H. Spanier. "Semigroups, Presburger formulas, and languages." In: *Pacific Journal of Mathematics* 16 (1966), pp. 285–296 (cit. on pp. 118, 119, 140).

[Gut76] John V. Guttag. "Abstract Data Types and the Development of Data Structures." In: *Proceedings of the SIGPLAN '76 Conference on Data: Abstraction, Definition and Structure, Salt Lake City, Utah, USA, March 22-24, 1976*. ACM, 1976, p. 72 (cit. on p. 53).

[Hac72] M. Hack. *Analysis of Production Schemata by Petri Nets*. Tech. rep. Cambridge, MA, USA, 1972 (cit. on pp. 24, 76, 97).

[Han+04] Marit Hansen et al. "Privacy-enhancing identity management." In: *Information Security Technical Report* 9.1 (2004), pp. 35–44 (cit. on p. 21).

[Hee+09] Kees M. van Hee et al. "Generation of Database Transactions with Petri Nets." In: *Fundam. Inform.* 93.1-3 (2009), pp. 171–184 (cit. on p. 59).

[HGD08] Monika Heiner, David R. Gilbert, and Robin Donaldson. "Petri Nets for Systems and Synthetic Biology." In: *Formal Methods for Computational Systems Biology, 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2008, Bertinoro, Italy, June 2-7, 2008, Advanced Lectures*. Ed. by Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro. Vol. 5016. Lecture Notes in Computer Science. Springer, 2008, pp. 215–264 (cit. on pp. 76, 97).

[HK97] Miki Hermann and Phokion G. Kolaitis. "On the Complexity of Unification and Disunification in Commutative Idempotent Semigroups." In: *Principles and Practice of Constraint Programming - CP97, Third International Conference, Linz, Austria, October 29 - November 1, 1997, Proceedings*. Ed. by Gert Smolka. Vol. 1330. Lecture Notes in Computer Science. Springer, 1997, pp. 282–296 (cit. on p. 55).

[Hoa72] C. A. R. Hoare. "Proof of correctness of data representations." In: *Acta Informatica* 1.4 (Dec. 1972), pp. 271–281 (cit. on pp. 89, 96).

[HR83] Theo Haerder and Andreas Reuter. "Principles of Transaction-oriented Database Recovery." In: *ACM Comput. Surv.* 15.4 (Dec. 1983), pp. 287–317 (cit. on p. 74).

[Hul+10] Richard Hull et al. "Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles." In: *Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers*. Ed. by Mario Bravetti and Tevfik Bultan. Vol. 6551. Lecture Notes in Computer Science. Springer, 2010, pp. 1–24 (cit. on p. 58).

[Hul08] Richard Hull. "Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges." In: *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part II*. 2008, pp. 1152–1163 (cit. on p. 58).

[IAU07] Hamidah Ibrahim, Ali Amer Alwan, and Nur Izura Udzir. "Checking Integrity Constraints with Various Types of Integrity Tests for Distributed Databases." In: *Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007), 3-6 December 2007, Adelaide, Australia*. Ed. by David S. Munro et al. IEEE Computer Society, 2007, pp. 151–152 (cit. on p. 76).

[Ibr06] Hamidah Ibrahim. "Checking Integrity Constraints - How it Differs in Centralized, Distributed and Parallel Databases." In: *17th International Workshop on Database and Expert Systems Applications (DEXA 2006), 4-8 September 2006, Krakow, Poland*. IEEE Computer Society, 2006, pp. 563–568 (cit. on p. 76).

[Jen81a] Kurt Jensen. "Coloured Petri Nets and the Invariant-Method." In: *Theor. Comput. Sci.* 14 (1981), pp. 317–336 (cit. on pp. 76, 97).

[Jen81b] Kurt Jensen. "How to Find Invariants for Coloured Petri Nets." In: *Mathematical Foundations of Computer Science 1981, Strbske Pleso, Czechoslovakia, August 31 - September 4, 1981, Proceedings*. Ed. by Jozef Gruska and Michal Chytil. Vol. 118. Lecture Notes in Computer Science. Springer, 1981, pp. 327–338 (cit. on pp. 76, 97).

[JK09] Kurt Jensen and Lars Michael Kristensen. *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*. Springer, 2009 (cit. on pp. 59, 98, 109, 190, 191).

[JK15] Kurt Jensen and Lars Michael Kristensen. "Colored Petri nets: a graphical language for formal modeling and validation of concurrent systems." In: *Commun. ACM* 58.6 (2015), pp. 61–70 (cit. on pp. 59, 190).

[JKV06] T. S. Jayram, Phokion G. Kolaitis, and Erik Vee. "The containment problem for REAL conjunctive queries with inequalities." In: *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*. Ed. by Stijn Vansummeren. ACM, 2006, pp. 80–89 (cit. on p. 54).

[Jün+10] Michael Jünger et al., eds. *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. Springer, 2010 (cit. on p. 139).

[KG14a] Yasir Imtiaz Khan and Nicolas Guelfi. "SLAPN : A Tool for Slicing Algebraic Petri Nets." In: *Proceedings of the International Workshop on Petri Nets and Software Engineering, co-located with 35th International Conference on Application and Theory of Petri Nets and Concurrency (PetriNets 2014) and 14th International Conference on Application of Concurrency to System Design (ACSD 2014), Tunis, Tunisia, June 23-24, 2014*. Ed. by Daniel Moldt and Heiko Rölke. Vol. 1160. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pp. 343–345 (cit. on p. 98).

[KG14b] Yasir Imtiaz Khan and Nicolas Guelfi. "Slicing High-level Petri Nets." In: *Proceedings of the International Workshop on Petri Nets and Software Engineering, co-located with 35th International Conference on Application and Theory of Petri Nets and Concurrency (PetriNets 2014) and 14th International Conference on Application of Concurrency to System Design (ACSD 2014), Tunis, Tunisia, June 23-24, 2014*. Ed. by Daniel Moldt and Heiko Rölke. Vol. 1160. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pp. 201–220 (cit. on p. 98).

[Kha13] Yasir Imtiaz Khan. "Optimizing Verification of Structurally Evolving Algebraic Petri Nets." In: *Software Engineering for Resilient Systems, 5th International Workshop, SERENE 2013, Kiev, Ukraine, October 3-4, 2013. Proceedings*. Ed. by Anatoliy Gorbenko, Alexander Romanovsky, and Vyacheslav S.

Kharchenko. Vol. 8166. Lecture Notes in Computer Science. Springer, 2013, pp. 64–78 (cit. on p. 98).

[Kha15] Yasir Imtiaz Khan. "Property based model checking of structurally evolving Algebraic Petri nets." PhD thesis. University of Luxembourg, 2015 (cit. on p. 98).

[Kin+97] Ekkart Kindler et al. "Petri Net Based Verification of Distributed Algorithms: An Example." In: *Formal Asp. Comput.* 9.4 (1997), pp. 409–424 (cit. on p. 56).

[KK02] Dimitris Karagiannis and Harald Kühn. "Metamodelling Platforms." In: *E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings*. Ed. by Kurt Bauknecht, A Min Tjoa, and Gerald Quirchmayr. Vol. 2455. Lecture Notes in Computer Science. Springer, 2002, p. 182 (cit. on p. 74).

[KN92] Deepak Kapur and Paliath Narendran. "Complexity of Unification Problems with Associative-Commutative Operators." In: *J. Autom. Reasoning* 9.2 (1992), pp. 261–288 (cit. on p. 55).

[Köh07] Michael Köhler. "The Reachability Problem for Object Nets." In: *Fundam. Inform.* 79.3-4 (2007), pp. 401–413 (cit. on p. 95).

[KR13] Yasir Imtiaz Khan and Matteo Risoldi. "Optimizing Algebraic Petri Net Model Checking by Slicing." In: *Joint Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'13) and the International Workshop on Modeling and Business Environments (ModBE'13), Milano, Italy, June 24 - 25, 2013*. Ed. by Daniel Moldt. Vol. 989. CEUR Workshop Proceedings. CEUR-WS.org, 2013, pp. 275–294 (cit. on p. 98).

[KS87] Bernd Krämer and Heinz-Wilhelm Schmidt. "Types and Modules for Net Specifications." In: *Concurrency and Nets: Advances in Petri Nets*. Ed. by Klaus Voss, Hartmann J. Genrich, and Grzegorz Rozenberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 269–286 (cit. on p. 56).

[KV98] Ekkart Kindler and Hagen Völzer. "Flexibility in Algebraic Nets." In: *Application and Theory of Petri Nets 1998, 19th International Conference, ICATPN '98, Lisbon, Portugal, June 22-26, 1998, Proceedings*. Ed. by Jörg Desel and Manuel Silva Suárez. Vol. 1420. Lecture Notes in Computer Science. Springer, 1998, pp. 345–364 (cit. on pp. 76, 97).

[Las16] Slawomir Lasota. "Decidability Border for Petri Nets with Data: WQO Dichotomy Conjecture." In: *Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings*. Ed. by Fabrice Kordon and Daniel Moldt. Vol. 9698. Lecture Notes in Computer Science. Springer, 2016, pp. 20–36 (cit. on pp. 23, 58, 96, 200).

[LMW07] Niels Lohmann, Peter Massuthe, and Karsten Wolf. "Operating guidelines for finite-state services." In: *Petri Nets and Other Models of Concurrency–ICATPN 2007*. Springer Berlin Heidelberg, 2007, pp. 321–341 (cit. on pp. 97, 202).

[LS07] Ruopeng Lu and Shazia Sadiq. "A Survey of Comparative Business Process Modeling Approaches." English. In: *Business Information Systems*. Ed. by Witold Abramowicz. Vol. 4439. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 82–94 (cit. on p. 57).

[Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996 (cit. on p. 21).

[Mat05] Yuri V. Matiyasevich. "Hilbert's Tenth Problem and Paradigms of Computation." In: *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005, Proceedings*. Ed. by S. Barry Cooper, Benedikt Löwe, and Leen Torenvliet. Vol. 3526. Lecture Notes in Computer Science. Springer, 2005, pp. 310–321 (cit. on p. 203).

[McC+11] R.M. McConnell et al. "Certifying algorithms." In: *Computer Science Review* 5.2 (2011), pp. 119–161 (cit. on p. 23).

[Mel17] Roxane Van Mellaert. "Optimal Design of Steel Structures according to the Eurocodes using Mixed-integer Linear Programming Methods,, ; Optimaal ontwerp van stalen structuren volgens de Eurocodes aan de hand van gemengd-discrete lineaire optimalisatiemethodes." PhD thesis. Katholieke Universiteit Leuven, Belgium, 2017 (cit. on p. 139).

[Mey+13] Andreas Meyer et al. "Modeling and Enacting Complex Data Dependencies in Business Processes." In: *Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*. Ed. by Florian Daniel, Jianmin Wang, and Barbara Weber. Vol. 8094. Lecture Notes in Computer Science. Springer, 2013, pp. 171–186 (cit. on p. 57).

[Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1967 (cit. on pp. 15, 23, 80, 200).

[MMW13] Rupak Majumdar, Roland Meyer, and Zilong Wang. "Static Provenance Verification for Message Passing Programs." In: *Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings*. Ed. by Francesco Logozzo and Manuel Fähndrich. Vol. 7935. Lecture Notes in Computer Science. Springer, 2013, pp. 366–387 (cit. on p. 96).

[MNS14] Elisa Marengo, Werner Nutt, and Ognjen Savkovic. "Towards a Theory of Query Stability in Business Processes." In: *Proceedings of the 8th Alberto Mendelzon Workshop on Foundations of Data Management, Cartagena de Indias, Colombia, June 4-6, 2014*. 2014 (cit. on pp. 58, 98).

[MR15] Marco Montali and Andrey Rivkin. "Formal Verification of Petri Nets with Names." In: *Web Services, Formal Methods, and Behavioral Types - 11th International Workshop, WS-FM 2014, Eindhoven, The Netherlands, September 11-12, 2014, and 12th International Workshop, WS-FM/BEAT 2015, Madrid, Spain, September 4-5, 2015, Revised Selected Papers*. Ed. by Thomas T. Hildebrandt et al. Vol. 9421. Lecture Notes in Computer Science. Springer, 2015, pp. 29–47 (cit. on p. 98).

[MR16a] Marco Montali and Andrey Rivkin. "DB-Nets: on The Marriage of Colored Petri Nets and Relational Databases." In: *CoRR* abs/1611.03680 (2016) (cit. on pp. 54, 59, 95).

[MR16b] Marco Montali and Andrey Rivkin. "Model checking Petri nets with names using data-centric dynamic systems." In: *Formal Asp. Comput.* 28.4 (2016), pp. 615–641 (cit. on p. 98).

[MR17] Marco Montali and Andrey Rivkin. "DB-Nets: On the Marriage of Colored Petri Nets and Relational Databases." In: *T. Petri Nets and Other Models of Concurrency* 12 (2017), pp. 91–118 (cit. on p. 95).

[MSW11] Andreas Meyer, Sergey Smirnov, and Mathias Weske. "Data in Business Processes." In: *EMISA Forum* 31.3 (2011), pp. 5–31 (cit. on p. 57).

[Mur89] T. Murata. "Petri nets: Properties, analysis and applications." In: *Proceedings of the IEEE* 77.4 (Apr. 1989), pp. 541–580 (cit. on pp. 76, 97).

[MV86] Gérard Memmi and Jacques Vautherin. "Analysing Nets by the Invariant Method." In: *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*. Ed. by Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg. Vol. 254. Lecture Notes in Computer Science. Springer, 1986, pp. 300–336 (cit. on pp. 76, 97).

[Nar96] Paliath Narendran. "Unification Modulo ACI + 1 + 0." In: *Fundam. Inform.* 25.1 (1996), pp. 49–57 (cit. on p. 55).

[Nat96a] M.B. Nathanson. *Additive Number Theory – Inverse Problems and the Geometry of Sumsets*. Graduate Texts in Mathematics. Springer New York, 1996 (cit. on p. 140).

[Nat96b] M.B. Nathanson. *Additive Number Theory – The Classical Bases*. Graduate Texts in Mathematics. Springer New York, 1996 (cit. on p. 140).

[Ord+09] Carlos Ordonez et al. "A Referential Integrity Browser for Distributed Databases." In: *12th International Workshop on the Web and Databases, WebDB 2009, Providence, Rhode Island, USA, June 28, 2009*. 2009 (cit. on p. 76).

[Orr98] Ken Orr. "Data Quality and Systems Theory." In: *Commun. ACM* 41.2 (Feb. 1998), pp. 66–71 (cit. on p. 74).

[ÖV11] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems, Third Edition*. Springer, 2011 (cit. on p. 56).

[Pap08] Michael P. Papazoglou. *Web Services - Principles and Technology*. Prentice Hall, 2008 (cit. on p. 202).

[Pap81] Christos H. Papadimitriou. "On the complexity of integer programming." In: *J. ACM* 28.4 (1981), pp. 765–768 (cit. on p. 139).

[PD12] Viara Popova and Marlon Dumas. "From Petri Nets to Guard-Stage-Milestone Models." In: *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*. Ed. by Marcello La Rosa and Pnina Soffer. Vol. 132. Lecture Notes in Business Information Processing. Springer, 2012, pp. 340–351 (cit. on p. 58).

[Pet62] Carl Adam Petri. "Kommunikation mit Automaten." ger. PhD thesis. Universität Hamburg, 1962 (cit. on p. 56).

[Pet86] C. A. Petri. "Concurrency Theory." In: *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*. Ed. by Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg. Vol. 254. Lecture Notes in Computer Science. Springer, 1986, pp. 4–24 (cit. on p. 56).

[Pop13] Louchka Popova-Zeugmann. *Time and Petri Nets*. Springer, 2013 (cit. on p. 95).

[Rei13] Wolfgang Reisig. *Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies*. Springer, 2013 (cit. on pp. 24, 25, 29, 30, 41, 43, 46, 54–57, 61, 69, 76, 90, 96, 97, 175, 200).

[Rei91] Wolfgang Reisig. "Petri Nets and Algebraic Specifications." In: *Theor. Comput. Sci.* 80.1 (1991), pp. 1–34 (cit. on pp. 24, 25, 29, 42, 43, 56, 76, 97).

[Rei98] Wolfgang Reisig. *Elements of distributed algorithms: modeling and analysis with Petri nets*. Springer, 1998 (cit. on pp. 21, 54, 56, 76, 97).

[RF10] Fernando Rosa-Velardo and David de Frutos-Escrig. "Decidability Problems in Petri Nets with Names and Replication." In: *Fundam. Inform.* 105.3 (2010), pp. 291–317 (cit. on pp. 23, 58, 200).

[RGN17] Veena Ravishankar, Kimberly A. Gero, and Paliath Narendran. *Asymmetric Unification and Disunification*. Tech. rep. 2017. arXiv: 1706.05066 (cit. on p. 55).

[Ric53] H. G. Rice. "Classes of Recursively Enumerable Sets and Their Decision Problems." In: *Transactions of the American Mathematical Society* 74.2 (1953), pp. 358–366 (cit. on p. 201).

[Rob65] J. A. Robinson. "A Machine-Oriented Logic Based on the Resolution Principle." In: *J. ACM* 12.1 (Jan. 1965), pp. 23–41 (cit. on pp. 55, 188).

[RS09] Pierre-Alain Reynier and Arnaud Sangnier. "Weak Time Petri Nets Strike Back!" In: *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*. Ed. by Mario Bravetti and Gianluigi Zavattaro. Vol. 5710. Lecture Notes in Computer Science. Springer, 2009, pp. 557–571 (cit. on p. 95).

[Sch86] Manfred Schmidt-Schauß. "Unification in Many-Sorted Eqational Theories." In: *8th International Conference on Automated Deduction, Oxford, England, July 27 - August 1, 1986, Proceedings*. Ed. by Jörg H. Siekmann. Vol. 230. Lecture Notes in Computer Science. Springer, 1986, pp. 538–552 (cit. on pp. 40, 55).

[Sch89] Manfred Schmidt-Schauß. "Unification in a Combination of Arbitrary Disjoint Equational Theories." In: *J. Symb. Comput.* 8.1/2 (1989), pp. 51–99 (cit. on pp. 40, 55).

[Sch97] Karsten Schmidt. "Siphons, Traps and High-Level Nets with Infinite Color Domains." In: *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings*. Ed. by Pierre Azéma and Gianfranco Balbo. Vol. 1248. Lecture Notes in Computer Science. Springer, 1997, pp. 271–289 (cit. on pp. 76, 97).

[Sch99] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999 (cit. on p. 139).

[SMN16] Ognjen Savkovic, Elisa Marengo, and Werner Nutt. "Query Stability in Monotonic Data-Aware Business Processes." In: *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*. Ed. by Wim Martens and Thomas Zeume. Vol. 48. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 16:1–16:18 (cit. on pp. 58, 98).

[ST12] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2012 (cit. on pp. 24, 30, 31, 33, 53, 71, 199, 200).

[ST13] Jan Sürmeli and Marvin Triebel. "Synthesizing Cost-Minimal Partners for Services." In: *Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings*. Ed. by Samik Basu et al. Vol. 8274. Lecture Notes in Computer Science. Springer, 2013, pp. 584–591 (cit. on pp. 97, 202).

[Sta84] Ryan Stansifer. *Presburger's Article on Integer Airthmetic: Remarks and Translation*. Tech. rep. TR84-639. Cornell University, Computer Science Department, Sept. 1984 (cit. on p. 139).

[SWZ05] Gopalan Sivathanu, Charles P. Wright, and Erez Zadok. "Ensuring Data Integrity in Storage: Techniques and Applications." In: *Proceedings of the 2005 ACM Workshop on Storage Security and Survivability*. StorageSS '05. Fairfax, VA, USA: ACM, 2005, pp. 26–36 (cit. on p. 74).

[TS15] Marvin Triebel and Jan Sürmeli. "Characterizing Stable Inequalities of Petri Nets." In: *Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings*. Ed. by Raymond R. Devillers and Antti Valmari. Vol. 9115. Lecture Notes in Computer Science. Springer, 2015, pp. 266–286 (cit. on pp. 76, 97).

[TS16a] Marvin Triebel and Jan Sürmeli. "Characterizing Stable and Deriving Valid Inequalities of Petri Nets." In: *Fundam. Inform.* 146.1 (2016), pp. 1–34 (cit. on pp. 76, 97).

[TS16b] Marvin Triebel and Jan Sürmeli. "Homogeneous Equations of Algebraic Petri Nets." In: *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*. Ed. by Josée Desharnais and Radha Jagadeesan. Vol. 59. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 14:1–14:14 (cit. on pp. 76, 84, 95, 97, 118).

[TS16c] Marvin Triebel and Jan Sürmeli. *Homogeneous Equations of Algebraic Petri Nets*. Tech. rep. 2016 (cit. on pp. 84, 95, 97, 118).

[Van+17] Thomas Vanhove et al. "Data transformation as a means towards dynamic data storage and polyglot persistence." In: *Int. Journal of Network Management* 27.4 (2017) (cit. on pp. 21, 22).

[Vau85] Jacques Vautherin. "Non-linear invariants for coloured Petri nets with interdependent tokens; application to the proof of parallel programs." In: *Advances in Petri Nets 1985, covers the 6th European Workshop on Applications and Theory in Petri Nets, Espoo, Finland in June 1985, selected papers*. Ed. by Grzegorz Rozenberg. Vol. 222. Lecture Notes in Computer Science. Springer, 1985, pp. 418–434 (cit. on pp. 61, 76, 97, 200).

[Vau86] Jacques Vautherin. "Parallel systems speciticications with coloured Petri nets and algebraic specifications." In: *Advances in Petri Nets 1987, covers the 7th European Workshop on Applications and Theory of Petri Nets, Oxford, UK, June 1986*. Ed. by Grzegorz Rozenberg. Vol. 266. Lecture Notes in Computer Science. Springer, 1986, pp. 293–308 (cit. on pp. 24, 25, 56).

[Vau87] Jacques Vautherin. "Calculation of Semi-Flows for Pr/T-Systems." In: *Proceedings of the Second International Workshop on Petri Nets and Performance Models, PNPM 1987, Madison, Wisconsin, USA, August 24-26, 1987*. IEEE Computer Society, 1987, pp. 174–183 (cit. on pp. 76, 97).

[Wag15] Christoph Wagner. "Partner datenverarbeitender Services." PhD thesis. Humboldt-Universität zu Berlin, 2015 (cit. on pp. 58, 95, 202).

[Wer11] Jan Martijn van der Werf. "Compositional design and verification of component-based information systems." PhD thesis. Eindhoven University of Technology, 2011 (cit. on p. 59).

[WKL14] Sebastian Wagner, Oliver Kopp, and Frank Leymann. "Choreography-based Consolidation of Multi-instance BPEL Processes." In: *CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain, April 3-5, 2014*. Ed. by Markus Helfert et al. SciTePress, 2014, pp. 287–298 (cit. on p. 57).

[Woh+05] Petia Wohed et al. "Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams." In: *Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings*. Ed. by Lois M. L. Delcambre et al. Vol. 3716. Lecture Notes in Computer Science. Springer, 2005, pp. 63–78 (cit. on p. 57).

[Woh+06] P. Wohed et al. "On the Suitability of BPMN for Business Process Modelling." English. In: *Business Process Management*. Ed. by Schahram Dustdar, JoséLuiz Fiadeiro, and AmitP. Sheth. Vol. 4102. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 161–176 (cit. on p. 57).

[Wol14] Karsten Wolf. "Explizites Model Checking: Welche Vorteile bieten Petrinetze?" In: *Informatik Spektrum* 37.3 (2014), pp. 220–228 (cit. on p. 56).

[Wyn+09] Moe Thandar Wynn et al. "Business process verification - finally a reality!" In: *Business Proc. Manag. Journal* 15.1 (2009), pp. 74–92 (cit. on p. 57).

SELBSTÄNDIGKEITSERKLÄRUNG

gemäß §7 Absatz 4 der Promotionsordnung von 2014 der Mathematisch-Naturwissenschaftlichen Fakultät der Humboldt-Universität zu Berlin

Ich erkläre, dass ich die Dissertation selbständig und nur unter Verwendung der von mir gemäß § 7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 126/2014 am 18.11.2014 angegebenen Hilfsmittel angefertigt habe.

Marvin Triebel